NVIDIA MLNX_OFED Documentation v23.10-7.1.8.0 LTS

Table of Contents

1 Overview

This is a long-term support (LTS) release. LTS is the practice of maintaining a software product for an extended period of time (up to three years) to help increase product stability. LTS releases include bug fixes and security patches.

NVIDIA® OpenFabrics Enterprise Distribution for Linux (MLNX_OFED) is a single Virtual Protocol Interconnect (VPI) software stack that operates across all NVIDIA network adapter solutions.

NVIDIA OFED (MLNX_OFED) is an NVIDIA-tested and packaged version of OFED and supports two interconnect types using the same RDMA (remote DMA) and kernel bypass APIs called OFED verbs—InfiniBand and Ethernet. Up to 400Gb/s InfiniBand and RoCE (based on the RDMA over Converged Ethernet standard) over 10/25/40/50/100/200/400Gb/s are supported with OFED to enable OEMs and System Integrators to meet the needs of end users in the said markets.

Further information on this product can be found in the following MLNX_OFED documents:

- (23.10-6-LTS6-2023Branch) Release Notes
- (23.10-6-LTS6-2023Branch) User Manual

1.1 Software Download

Please visit nvidia.com/en-us/networking → Products → Software → InfiniBand Drivers → NVIDIA MLNX_OFED

1.2 Document Revision History

For the list of changes made to the User Manual, refer to (23.10-6-LTS6-2023Branch) User Manual Revision History.

For the list of changes made to the Release Notes, refer to Release Notes History.

2 Release Notes

2.1 Release Notes Update History

Version	Date	Description
23.10-7.1.8.0	May 2026	Initial release of this document version. This release introduces Bug Fixes in This Version.

As of MLNX_OFED version v5.1-0.6.6.0, the following are no longer supported.

- ConnectX-3
- ConnectX-3 Pro
- Connect-IB
- RDMA experimental verbs libraries (mlnx_lib)

To utilize the above devices/libraries, refer to version 4.9 long-term support (LTS).

Release Notes contain the following sections:

- General Support
- Changes and New Features
- Bug Fixes in This Version
- Known Issues

2.2 Supported NIC Speeds

The Linux Driver operates across all NVIDIA network adapter solutions supporting the following uplinks to servers:

Uplink/Adapter Card	Driver Name	Uplink Speed
BlueField-2	mlx5	• InfiniBand: SDR, FDR, EDR, HDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE, 100GbE
BlueField		• InfiniBand: SDR, QDR, FDR, FDR10, EDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE, 100GbE
ConnectX-7		• InfiniBand: EDR, HDR100, HDR, NDR200, NDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE, 100GbE, 200GbE, 400GbE
ConnectX-6 Lx		• Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE
ConnectX-6 Dx		• Ethernet: 10GbE, 25GbE, 40GbE, 50GbE, 100GbE, 200GbE
ConnectX-6		• InfiniBand: SDR, FDR, EDR, HDR • Ethernet: 10GbE, 25GbE, 40GbE, 50GbE, 100GbE, 200GbE
ConnectX-5/ConnectX-5 Ex		• InfiniBand: SDR, QDR, FDR, FDR10, EDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE, 100GbE
ConnectX-4 Lx		• Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE

Uplink/Adapter Card	Driver Name	Uplink Speed
ConnectX-4		• InfiniBand: SDR, QDR, FDR, FDR10, EDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE, 56GbE, 100Gb

Important Notes:

- 50GbE and 100GbE are speed that supports both NRZ and PAM4 modes in Force mode and Auto-Negotiation mode.
- 200GbE is a speed that supports PAM4 mode only.
- 56GbE is an NVIDIA proprietary link speed and can be achieved while connecting an NVIDIA adapter card to NVIDIA SX10XX switch series or when connecting an NVIDIA adapter card to another NVIDIA adapter card.

2.3 Embedded Components

Package	Revision	Licenses
clusterkit	1.11.442-1.2310055	BSD
dpcp	1.1.43-1.2310055	BSD-3-Clause
hcoll	4.8.3228-1.20250521.6f14f256	Proprietary
ibarr	0.1.3-1.2310055	(GPL-2.0 WITH Linux-syscall-note) OR BSD-2-Clause
ibdump	6.0.0-1.2310055	BSD2+GPL2
ibsim	0.12-1.2310055	GPLv2 or BSD
ibutils2	2.1.1-0.21501.MLNX20240610.g840ec16f.2310615	Mellanox Confidential and Proprietary
iser	23.10-OFED.23.10.6.1.5.1	GPLv2
isert	23.10-OFED.23.10.6.1.5.1	GPLv2
kernel-mft	4.26.1-35	Dual BSD/GPL
knem	1.1.4.90mlnx3-OFED.23.10.0.2.1.1	BSD and GPLv2
libvma	9.8.40-1	GPLv2 or BSD
libxlio	3.20.8-1	GPLv2 or BSD
mlnx-en	23.10-6.1.5.0.g2f2078a	GPLv2
mlnx-ethtool	6.4-1.2310055	GPL
mlnx-iproute2	6.4.0-1.2310055	GPL
mlnx-nfsrdma	23.10-OFED.23.10.6.1.5.1	GPLv2
mlnx-nvme	23.10-OFED.23.10.6.1.5.1	GPLv2
mlnx-ofa_kernel	23.10-OFED.23.10.6.1.5.1	GPLv2
mlnx-tools	23.10-0.2310409	GPLv2 or BSD

Package	Revision	Licenses
mlx-steering-dump	1.0.0-0.2310055	GPLv2
mpitests	3.2.21-8418f75.2310055	BSD
mstflint	4.16.1-2.2310055	GPL/BSD
multiperf	3.0-3.0.2310055	BSD 3-Clause, GPL v2 or later
ofed-docs	23.10-OFED.23.10.6.1.5	GPL/BSD
ofed-scripts	23.10-OFED.23.10.6.1.5	GPL/BSD
openmpi	4.1.7a1-1.2310055	BSD
opensm	5.17.2.MLNX20240610.dc7c2998-0.1.2310409	GPLv2 or BSD
openvswitch	2.17.8-1.2310409	ASL 2.0 and LGPLv2+ and SISSL
perftest	23.10.0-0.96.g9729d3d.2310615	BSD 3-Clause, GPL v2 or later
rdma-core	2307mlnx47-1.2310409	GPLv2 or BSD
rshim	2.0.19-0.gbf7f1f2	GPLv2
sharp	3.5.2.MLNX20240610.61bf97b7-1.2310615	Proprietary
sockperf	3.10-0.git5ebd327da983.2310055	BSD
srp	23.10-OFED.23.10.6.1.5.1	GPLv2
ucx	1.16.0-1.2310409	BSD
xpmem	2.7.3-1.2310055	GPLv2 and LGPLv2.1
xpmem-lib	2.7-0.2310055	LGPLv2.1

2.4 General Support

Supported Operating Systems

Operating System	Architecture	Default Kernel Version (Primary)/ Tested with Kernel Version (Community)	OS Support Model	ASAP^2 OVS-Kernel SR-IOV	ASAP^2 OVS-DPDK SR-IOV	NFS over RDMA	NVMe	GPUDirect Storage (GDS)	UCX - CUDA Version
Alma 8.5	x86_64	4.18.0-348.12.2.EL8_5.X86_64	Community	✗	✗	✗	✗	✗	✗
Anolis OS 3.2	x86_64	5.10.134-13.al8.x86_64	Community	✗	✗	✗	✗	✗	✗
Anolis OS 8.6	AArch64	5.10.134+	Primary	✗	✗	✗	✗	✗	✗
	x86_64	5.10.134+	Primary	✗	✗	✗	✗	✗	✗
BCLINUX2 1.10SP2	AArch64	4.19.90-2107.6.0.0098.oe1.bclinux.aarch64	Primary	✗	✗	✗	✓	✗	✗

	x86_64	4.19.90-2107.6.0.0100.oe1.bclinux.x86_64	Primary	✗	✗	✓	✓	✗	✗
CentOS Stream v8	AArch64	4.18.0-553.el8.aarch64	Community	✗	✗	✗	✗	✗	✗
	x86_64	4.18.0-553.el8.x86_64	Community	✗	✗	✗	✗	✗	✗
CentOS Stream v9	AArch64	5.14.0-419.el9.x86_64	Community	✗	✗	✗	✗	✗	✗
	x86_64	5.14.0-419.el9.aarch64	Community	✗	✗	✗	✗	✗	✗
CTYUNOS 2.0	AArch64	4.19.90-2102.2.0.0062.ctl2.aarch64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	4.19.90-2102.2.0.0062.ctl2.x86_64	Primary	✗	✗	✗	✗	✗	✗
CTYUNOS 23.01	AArch64	5.10.0-136.12.0.86.ctl3.aarch64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	5.10.0-136.12.0.86.ctl3.x86_64	Primary	✗	✗	✗	✗	✗	✗
Debian9.13	AArch64	4.9.0-13-arm64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	4.9.0-13-amd64	Primary	✗	✗	✗	✗	✗	✗
Debian10.8	AArch64	4.19.0-14-arm64	Primary	✗	✗	✗	✗	✗	✗
Debian10.9	x86_64	4.19.0-14-amd64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	4.19.0-16-amd64	Primary	✓	✗	✗	✓	✗	✗
Debian10.13	AArch64	4.19.0-21-arm64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	4.19.0-21-amd64	Primary	✗	✗	✗	✗	✗	✗
Debian11.3	AArch64	5.10.0-13-arm64	Primary	✓	✗	✗	✓	✗	✗
	x86_64	5.10.0-13-amd64	Primary	✓	✗	✗	✓	✗	✗
Debian12	AArch64	6.1.0-10-arm64	Primary	✓	✗	✗	✓	✗	✗
	x86_64	6.1.0-10-amd64	Primary	✓	✗	✗	✓	✗	✗
Debian12.5	AArch64	6.1.0-18-arm64	Primary	✓	✗	✗	✓	✗	✗
	x86_64	6.1.0-18-amd64	Primary	✓	✗	✗	✓	✗	✗

EulerOS2.0sp9	AArch64	4.19.90-vhulk2006.2.0.h171.eulerosv2r9.aarch64	Community	✗	✗	✗	✗	✗	✗
	x86_64	4.18.0-147.5.1.0.h269.eulerosv2r9.x86_64	Community	✗	✗	✗	✗	✗	✗
EulerOS2.0sp10	AArch64	4.19.90-vhulk2110.1.0.h860.eulerosv2r10.aarch64	Community	✗	✗	✗	✗	✗	✗
	x86_64	4.18.0-147.5.2.4.h694.eulerosv2r10.x86_64	Community	✗	✗	✗	✗	✗	✗
EulerOS2.0sp11	AArch64	5.10.0-60.18.0.50.h323.eulerosv2r11.aarch64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	5.10.0-60.18.0.50.h323.eulerosv2r11.x86_64	Primary	✗	✗	✗	✗	✗	✗
EulerOS2.0sp12	AArch64	5.10.0-136.12.0.86.h1032.eulerosv2r12.aarch64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	5.10.0-136.12.0.86.h1032.eulerosv2r12.x86_64	Primary	✗	✗	✗	✗	✗	✗
KYLIN10SP2	AArch64	4.19.90-24.4.v2101.ky10.aarch64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	4.19.90-24.4.v2101.ky10.x86_64	Primary	✓	✗	✗	✗	✗	✗
KYLIN10SP3	AArch64	4.19.90-52.15.v2207.ky10.aarch64	Primary	✓	✗	✗	✗	✗	✗
	x86_64	4.19.90-52.15.v2207.ky10.x86_64	Primary	✓	✗	✗	✗	✗	✗
Mariner 2.0	x86_64	5.15.118.1-1.cm2.x86_64	Community	✗	✗	✗	✗	✗	✗
Oracle Linux 7.9	x86_64	5.4.17-2011.6.2.el7uek.x86_64	Primary	✗	✗	✗	✗	✗	✗
Oracle Linux 8.4	x86_64	5.4.17-2102.201.3.el8uek.x86_64	Primary	✗	✗	✗	✗	✗	✗
Oracle Linux 8.6	x86_64	5.4.17-2136.307.3.1.el8uek.x86_64	Primary	✗	✗	✗	✗	✗	✗
Oracle Linux 8.7	x86_64	5.15.0-3.60.5.1.el8uek.x86_64	Primary	✗	✗	✗	✗	✗	✗
Oracle Linux 8.8	x86_64	5.15.0-101.103.2.1.el8uek.x86_64	Primary	✗	✗	✗	✗	✗	✗
Oracle Linux 9.0	x86_64	5.15.0-0.30.19.el9uek.x86_64	Primary	✗	✗	✗	✗	✗	✗

Oracle Linux 9.1	x86_64	5.15.0-3.60.5.1.el9uek.x86_64	Primary	✗	✗	✗	✗	✗	✗
Oracle Linux 9.2	x86_64	5.15.0-101.103.2.1.el9uek.x86_64	Primary	✗	✗	✗	✗	✗	✗
OpenSUSE 15.3	AArch64	-	Community	✗	✗	✗	✗	✗	✗
	x86_64	5.3.18-150300.59.43-DEFAULT	Community	✗	✗	✗	✗	✗	✗
OPENEULER R20.03SP1	AArch64	4.19.90-2012.4.0.0053.OE1.AARCH64	Community	✗	✗	✗	✗	✗	✗
	x86_64	4.19.90-2110.8.0.0119.OE1.X86_64	Community	✗	✗	✗	✗	✗	✗
OPENEULER R20.03SP3	AArch64	4.19.90-2112.8.0.0131.oe1.aarch64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	4.19.90-2112.8.0.0131.oe1.x86_64	Primary	✗	✗	✗	✗	✗	✗
OPENEULER R22.03	AArch64	5.10.0-60.18.0.50.oe2203.aarch64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	5.10.0-60.18.0.50.oe2203.x86_64	Primary	✗	✗	✗	✗	✗	✗
Photon OS 3.0	x86_64	4.19.225-3.ph3	Community	✗	✗	✗	✗	✗	✗
RHEL/ CentOS7.2	x86_64	3.10.0-327.el7.x86_64	Primary	✗	✗	✗	✓	✗	12.2
RHEL/ CentOS7.4	x86_64	3.10.0-693.el7.x86_64	Primary	✓	✓	✗	✓	✗	12.2
RHEL/ CentOS7.6	x86_64	3.10.0-957.el7.x86_64	Primary	✓	✓	✓	✓	✗	12.2
RHEL/ CentOS7.6 alternate	aarch64	4.14.0-115.el7a.aarch64	Community	✓	✓	✗	✗	✗	✗
RHEL/ CentOS7.7	x86_64	3.10.0-1062.el7.x86_64	Primary	✓	✓	✗	✗	✗	12.2
RHEL/ CentOS7.8	x86_64	3.10.0-1127.el7.x86_64	Primary	✓	✓	✓	✓	✗	12.2
RHEL/ CentOS7.9	x86_64	3.10.0-1160.el7.x86_64	Primary	✓	✓	✓	✓	✗	12.2
RHEL/ CentOS8.0	AArch64	4.18.0-80.el8.aarch64	Primary	✓	✓	✗	✗	✓	12.2
	x86_64	4.18.0-80.el8.x86_64	Primary	✓	✓	✗	✗	✓	12.2
RHEL/ CentOS8.1	AArch64	4.18.0-147.el8.aarch64	Primary	✓	✓	✗	✗	✓	12.2
	x86_64	4.18.0-147.el8.x86_64	Primary	✓	✓	✗	✗	✓	12.2

RHEL/ CentOS8.2	AArch64	4.18.0-193.el8.aarch64	Primary	✓	✓	✗	✓	✓	12.2
	x86_64	4.18.0-193.el8.x86_64	Primary	✓	✓	✓	✓	✓	12.2
RHEL/ CentOS8.3	AArch64	4.18.0-240.el8.aarch64	Primary	✓	✓	✗	✗	✓	12.2
	x86_64	4.18.0-240.el8.x86_64	Primary	✓	✓	✗	✗	✓	12.2
RHEL/ CentOS8.4	AArch64	4.18.0-305.el8.aarch64	Primary	✓	✓	✗	✓	✓	12.2
	x86_64	4.18.0-305.el8.x86_64	Primary	✓	✓	✓	✓	✓	12.2
RHEL/ CentOS/ Rocky8.5	AArch64	4.18.0-348.el8.aarch64	Primary	✓	✓	✗	✓	✓	12.2
	x86_64	4.18.0-348.el8.x86_64	Primary	✓	✓	✓	✓	✓	12.2
RHEL/ Rocky8.6	AArch64	AArch644.18.0-372.41.1.el8_6.aarch64	Primary	✓	✓	✗	✓	✓	12.2
	x86_64	4.18.0-372.41.1.el8_6.x86_64	Primary	✓	✓	✓	✓	✓	12.2
RHEL/ Rocky8.7	AArch64	4.18.0-425.14.1.el8_7.aarch64	Primary	✓	✗	✗	✓	✓	12.2
	x86_64	4.18.0-425.14.1.el8_7.x86_64	Primary	✓	✗	✓	✓	✓	12.2
RHEL/ Rocky8.8	AArch64	4.18.0-477.10.1.el8_8.aarch64	Primary	✓	✗	✗	✓	✓	12.2
	x86_64	4.18.0-477.10.1.el8_8.x86_64	Primary	✓	✗	✓	✓	✓	12.2
RHEL/ Rocky8.9	AArch64	4.18.0-513.5.1.el8_9.aarch64	Primary	✓	✗	✗	✓	✓	12.2
	x86_64	4.18.0-513.5.1.el8_9.x86_64	Primary	✓	✗	✓	✓	✓	12.2
RHEL/ Rocky8.10	AArch64	4.18.0-553.el8_10.aarch64	Primary	✓	✗	✗	✓	✓	12.2
	x86_64	4.18.0-553.el8_10.x86_64	Primary	✓	✗	✓	✓	✓	12.2
RHEL/ Rocky9.0	AArch64	5.14.0-70.46.1.el9_0.aarch64	Primary	✓	✗	✗	✓	✓	12.2
	x86_64	5.14.0-70.46.1.el9_0.x86_64	Primary	✓	✗	✓	✓	✓	12.2
RHEL/ Rocky9.1	AArch64	5.14.0-162.19.1.el9_1.aarch64	Primary	✓	✗	✗	✓	✓	12.2
	x86_64	5.14.0-162.19.1.el9_1.x86_64	Primary	✓	✗	✓	✓	✓	12.2
RHEL/ Rocky9.2	AArch64	5.14.0-284.11.1.el9_2.aarch64	Primary	✓	✗	✗	✓	✓	12.2

	x86_64	5.14.0-284.11.1.el9_2.x86_64	Primary	✓	✗	✓	✓	✓	12.2
RHEL/ Rocky9.3	AArch64	5.14.0-362.8.1.el9_3.aarch64	Primary	✓	✗	✗	✓	✓	12.2
	x86_64	5.14.0-362.8.1.el9_3.x86_64	Primary	✓	✗	✓	✓	✓	12.2
RHEL/ Rocky9.4	AArch64	5.14.0-427.13.1.el9_4.aarch64	Primary	✓	✗	✗	✓	✓	12.2
	x86_64	5.14.0-427.13.1.el9_4.x86_64	Primary	✓	✗	✓	✓	✓	12.2
SLES12.1SP2	x86_64	4.4.21-69-default	Community	✗	✗	✗	✗	✗	✗
SLES12SP3	x86_64	4.4.73-5-default	Community	✗	✗	✗	✗	✗	✗
SLES12SP4	AArch64	4.12.14-94.41-default	Community	✓	✗	✗	✓	✗	✗
	x86_64	4.12.14-94.41-default	Community	✓	✗	✓	✓	✗	✗
SLES12SP5	AArch64	4.12.14-120-default	Primary	✓	✗	✗	✓	✗	✗
	x86_64	4.12.14-120-default	Primary	✓	✗	✓	✓	✗	✗
SLES15SP2	AArch64	5.3.18-22-default	Primary	✓	✓	✗	✓	✗	✗
	x86_64	5.3.18-22-default	Primary	✓	✓	✓	✓	✗	✗
SLES15SP3	AArch64	5.3.18-57-default	Primary	✓	✗	✗	✓	✗	✗
	x86_64	5.3.18-57-default	Primary	✓	✗	✓	✓	✗	✗
SLES15SP4	AArch64	5.14.21-150400.22-default	Primary	✓	✗	✗	✓	✗	✗
	x86_64	5.14.21-150400.22-default	Primary	✓	✗	✓	✓	✗	✗
SLES15SP5	AArch64	5.14.21-150500.53-default	Primary	✓	✗	✗	✓	✗	✗
	x86_64	5.14.21-150500.53-default	Primary	✓	✗	✓	✓	✗	✗
SLES15SP6	x86_64	6.4.0-150600.21-default	Primary	✓	✗	✓	✓	✗	✗
Ubuntu16.04	x86_64	4.4.0-21-generic	Community	✗	✗	✗	✗	✗	✗
Ubuntu18.04	AArch64	4.15.0-20-generic	Primary	✓	✓	✗	✓	✓	11.6
	x86_64	4.15.0-20-generic	Primary	✓	✓	✓	✓	✓	11.6

Ubuntu20.04	AArch64	5.4.0-26-generic	Primary	✓	✓	✗	✓	✓	12.2
	x86_64	5.4.0-26-generic	Primary	✓	✓	✓	✓	✓	12.2
Ubuntu22.04	AArch64	5.15.0-25-generic	Primary	✓	✓	✓	✓	✓	12.2
	x86_64	5.15.0-25-generic	Primary	✓	✓	✓	✓	✓	12.2
Ubuntu23.04	x86_64	6.2.0-20-generic	Primary	✓	✗	✓	✓	✗	✗
Ubuntu23.10	x86_64	6.5.0-5-generic	Primary	✓	✗	✗	✗	✗	✗
UOS20.1020	AArch64	4.19.90-2109.1.0.0108.up2.uel20.aarch64	Primary	✗	✗	✗	✗	✗	✗
	x86_64	4.19.90-2109.1.0.0108.up2.uel20.x86_64	Primary	✗	✗	✗	✗	✗	✗
UOS20.1040	AArch64	4.19.0-arm64-server	Primary	✗	✗	✗	✗	✗	✗
	x86_64	4.19.0-server-amd64	Primary	✗	✗	✗	✗	✗	✗
Citrix XenServer Host8.2	x86_64	4.19.0+1	Primary	✗	✗	✗	✗	✗	✗
Kernel 6.6	AArch64	6.6	Primary	✓	✗	✗	✓	✗	✗
	x86_64	6.6	Primary	✓	✗	✓	✓	✗	✗

⚠

- 32 bit platforms are no longer supported in MLNX_OFED
- For RPM-based distributions, to install OFED on a different kernel, create a new ISO image using mlnx_add_kernel_support.sh script (see the MLNX_OFED User Manual for instructions)
- Upgrading MLNX_OFED on a cluster requires upgrading all of its nodes to the newest version as well
- If using MLNX_OFED 4.9 LTS with MLNX_OFED 5.x with upstream verbs, MLNX_OFED 4.9 must be installed with `--upstream-libs` flag so the verbs libraries match.
- A combination of 4.9 LTS default verbs and MOFED 5.x upstream verbs is not supported.
- All operating systems listed above are fully supported in Paravirtualized and SR-IOV environments with Linux KVM Hypervisor

2.4.1 Upgrade/Downgrade Matrix

This section reflects which versions were tested and verified for upgrade and downgrade.

Target Version	Versions Verified for Upgrade/ Downgrade	Release Type	Release Date
23.10-7.1.8.0 LTS (December 2025)	23.10-6.1.6.1 • MLNX_OFED and DOCA-OFED Profile	LTS-U6	December 2025
	23.10-0.5.5.0 - MLNX_OFED and DOCA-OFED Profile	GA-LTS-U0	October 2023
	5.8-7.0.6.1	LTS-U7	June 2025
	5.4-3.7.5.0	GA-LTS-Update	November 2022

2.4.2 MLNX_OFED Version Interoperability

This section reflects which versions were tested and verified for multi-version environments.

Target Version	Verified OFED Version Interoperability	Release Type	Release Date
23.10-7.1.8.0 LTS (December 2025)	23.10-6.1.6.1 • MLNX_OFED and DOCA-OFED Profile	LTS-U6	December 2025
	5.8-7.0.6.1	LTS-U7	June 2025

2.4.3 Supported NIC Firmware Versions

As of version 5.1, ConnectX-3, ConnectX-3 Pro or Connect-IB adapter cards are no longer supported. To work with a version that supports these adapter cards, please refer to version 4.9 long-term support (LTS).

This current version is tested with the following NVIDIA adapter card firmware versions:

Adapter Card	Bundled Firmware Version
BlueField®-2	24.39.5172
ConnectX-7	28.39.5172
ConnectX-6 Lx	26.39.5172
ConnectX-6 Dx	22.39.5172
ConnectX-6	20.39.5172
ConnectX-5/ConnectX-5 Ex	16.35.4506
BlueField	18.33.1048
ConnectX-4	12.28.2006
ConnectX-4 Lx	14.32.1010

For the official firmware versions, please see https://www.nvidia.com/en-us/networking/ → Support → Support → Firmware Download.

2.5 Changes and New Features

2.5.1 New Features

The following are the new features and changes that were added in this version. The supported adapter cards are specified as follows:

Supported Cards	Description
All HCAs	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-4/ConnectX-4 Lx/ConnectX-5/ConnectX-6/ConnectX-6 Dx/ConnectX-6 Lx/ConnectX-7/BlueField-2
ConnectX-6 Dx and above	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-6 Dx/ConnectX-6 Lx/ConnectX-7/BlueField-2
ConnectX-6 and above	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-6/ConnectX-6 Dx/ConnectX-6 Lx/ConnectX-7/BlueField-2
ConnectX-5 and above	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-5/ConnectX-6/ConnectX-6 Dx/ConnectX-6 Lx/ConnectX-7/BlueField-2
ConnectX-4 and above	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-4/ConnectX-4 Lx/ConnectX-5/ConnectX-6/ConnectX-6 Dx/ConnectX-6 Lx/ConnectX-7/BlueField-2

Feature/Change	Description
23.10-7.1.8.0	
General	
Embedded Components	Updated the versions of the following embedded component: • MFT • ConnectX adapter cards firmware For further information, see Embedded Components section.
General	Bug fixes

For a list of features from previous versions, see Release Notes Change Log History section. For additional information on the new features, please refer to the MLNX_OFED User Manual.

2.5.2 API Changes in MLNX_OFED

2.5.2.1 MLNX_OFED Verbs API Migration

As of MLNX_OFED v5.0 release (Q1 of the year 2020), MLNX_OFED Verbs API have migrated from the legacy version of user space verbs libraries (libibervs, libmlx5, etc.) to the Upstream version rdma-core.

For the list of MLNX_OFED verbs APIs that have been migrated, refer to Migration to RDMA-Core document.

2.5.3 Customer Affecting Changes

2.5.3.1 Changes in This Release

This section provides a list of changes that took place in the current version and break compatibility/interface, discontinue support for features and/or OS versions, etc.

Introduced in Version	Description
23.10-6.1.6.1	XenServer 7.1 OS will not be supported in MLNX_OFED v23.10-6.x.x.x. The last update release that supports this OS is 23.10-5.1.4.0.

2.5.3.2 Changes Planned for Future Releases

This section provides a list of changes that will take place in a future version of the product and will break compatibility/interface, discontinue support for features and/or OS versions, etc.

Planned for Version	Description
N/A	N/A

2.5.3.3 Changes in Earlier Releases

This section provides a list of changes that took place throughout the past two major releases that broke compatibility/interface, discontinued support for features and/or OS versions, etc.

For an archive of all changes, please refer to the Release Notes History section.

Introduced in Version	Description	Customer Impact and Recommendation
N/A	N/A	N/A

2.5.3.4 Discontinued Features

List of features which are supported in previous generations of hardware devices.

N/A

2.5.4 Unsupported Features

This section provides a list of features that are not supported by the software.

- Soft-RoCE
- RDMA experimental verbs library (mlnx_lib)
- CIFS (Common Internet File System) module installation
- Relational Database Service (RDS)
- mthca InfiniBand driver
- Ethernet IPoIB (eIPoIB)

2.6 Bug Fixes in This Version

Below are the bugs fixed in this version. For a list of fixes previous version, see Bug Fixes History.

Internal Reference Number	Description
4921892 / 4913956	**Description:** Fixed an issue that occurred when CONFIG_UBSAN=y and the user provided an invalid value, resulting in the following UBSAN error:
	Keywords: CONFIG_UBSAN
	Discovered in Release: 23.10-6.1.6.1
	Fixed in Release: 23.10-7.1.8.0
4858694 / 4858587	**Description:** Fixed soft lockup warnings that could occur while releasing large UMEMs by periodically yielding the CPU during the release flow.
	Keywords: Soft lockup warnings
	Discovered in Release: 23.10-5.1.4.0
	Fixed in Release: 23.10-7.1.8.0

2.7 Known Issues

The following is a list of general limitations and known issues of the current version of the release.

Internal Ref. Number	Issue
3546668	**Description:** On 64k page size systems, applications that open a large number of RDMA resources (UARs/QPs/CQs etc.) might face errors creating those resources due to a PCI BAR size limitation.
	Keywords: PCI BAR size limitation
	Workaround: It is recommended to increase the BAR size via mlxconfig to allow enough space for the allocation of all the needed RDMA resources.
	Discovered in Release: 23.10-1.1.9.0
3678715	**Description:** When attempting to restart drivers using openIbd service while the nvme_rdma module is loaded, the process may fail. This behavior is intentional, as unloading nvme_rdma during the driver restart can lead to connectivity issues in other applications within the setup.
	Keywords: openIbd service, nvme_rdma module
	Workaround: Manually unload the nvme_rdma module before performing the driver restart. This can be achieved using the `modprobe -r nvme_rdma` command.
	Discovered in Release: 23.10-1.1.9.0
3676223	**Description:** When using kernel version 4.12 or above, it is advised to run echo 0 > /sys/bus/pci/devices/0000\:08\:00.0/sriov_drivers_autoprobe to avoid VF probing

Internal Ref. Number	Issue
	Keywords: VF probing
	Workaround: N/A
	Discovered in Release: 23.10-1.1.9.0
3682658	**Description:** While using the RDMA-CM user application and the AF_IB parameter, the kernel uses only the first byte of the private data to set the CMA version. In such scenario, any user data written to this byte will be overwritten.
	Keywords: RDMA-CM user application, AF_IB, private data
	Workaround: Do not use AF_IB for application's private data.
	Discovered in Release: 23.10-0.5.5.0
3640082	**Description:** A potential null pointer dereference might occur due to a missing update in the PCI subsystem code when creating the maximum number of VFs. All kernel versions lacking the following fix are impacted: "PCI: Avoid enabling PCI atomics on VFs."
	Keywords: Maximal VF number
	Workaround: N/A
	Discovered in Release: 23.10-0.5.5.0
3653417	**Description:** When offloading IPsec policy rules while in legacy mode there are two options: 1. Software steering - The software stack will handle the task, and no device offload will take place. 2. Changing the steering mode to firmware steering will return unsupported.
	Keywords: IPsec, legacy mode
	Workaround: Perform a devlink reload after changing the steering mode.
	Discovered in Release: 23.10-0.5.5.0
3612274	**Description:** Currently, either IPsec offload or TC offload for a specific interface is allowed. The offloading TC rule to an interface will fail if an IPSec rule is already offloaded on it, and vice-versa.
	Keywords: IPsec offload, TC offload
	Workaround: N/A
	Discovered in Release: 23.10-0.5.5.0
3596126	**Description:** OVS mirroring of both egress and ingress together with modified TTL is not supported by Connectx-5 cards, and may cause packets checksum issues and errors in the dmesg command.
	Keywords: OVS mirroring, Connectx-5
	Workaround: N/A
	Discovered in Release: 23.10-0.5.5.0
3538463	**Description:** A Kernel ABI problem in Sles15SP4 may lead to issues during driver start. This impacts kernels starting from version 5.14.21-150400.24.11.1 up to version 5.14.21-150400.24.63.1 (July 2022 to May 2023), inclusive. For more information, see https://www.suse.com/support/kb/doc/?id=000021137.
	Keywords: Kernel ABI, Sles15SP4, driver start

Internal Ref. Number	Issue
	Workaround: Upgrade to a kernel version newer than 5.14.21-150400.24.63.1 (May 2023).
	Discovered in Release: 23.10-0.5.5.0
3637252	**Description:** When running over REHL7.6 with excessive RDMA/RoCE workload, kernel warnings may be triggered.
	Keywords: REHL7.6, RDMA, RoCE
	Workaround: N/A
	Discovered in Release: 23.10-0.5.5.0

Internal Ref. Number	Issue
3046655	**Description:** A package manager upgrade with zypper (on an SLES system) may prompt a question about vendor change from "Mellanox Technologies" to "OpenFabrics".
	Keywords: Installation, SLES
	Workaround: Either accept the prompted change, or add the /etc/zypp/vendors.d/mlnx_ofed file with the following content: [main] vendors = Mellanox,OpenFabrics
	Discovered in Release: 23.07-0.5.0.0
3392477	**Description:** The ConnectX-7 firmware embedded in this MLNX_OFED version cannot be burnt using the MLNX_OFED installer script.
	Keywords: ConnectX-7, MLNX_OFED installer script
	Workaround: Please download and install the dedicated firmware from the web https://network.nvidia.com/support/firmware/connectx7ib/
	Discovered in Release: 23.07-0.5.0.0
3532756	**Description:** The kernel may crash when restarting the driver while IP sec rules are configured.
	Keywords: IP sec
	Workaround: Flush the IP sec configuration before reloading the driver: ip xfrm state flush ip xfrm policy flush
	Discovered in Release: 23.07-0.5.0.0
3472979	**Description:** When a large number of virtual functions are present, the output of the `"ip link show"` command may be truncated.
	Keywords: virtual functions, ip link show
	Workaround: N/A
	Discovered in Release: 23.07-0.5.0.0
3413938	**Description:** When using the mlnx-sf script, creating and deleting an SF with the same ID number in a stressful manner may cause the setup to hang due to a race between the create and delete commands.

Internal Ref. Number	Issue
	Keywords: Hang; mlnx-sf
	Workaround: N/A
	Discovered in Release: 23.07-0.5.0.0
3461572	**Description:** Configuring Multiport Eswitch LAG mode can be performed only via devlink from this release onwards. The compat sysfs should not be used to configure mpesw LAG.
	Keywords: Multiport Eswitch, compat sysfs, mpesw LAG
	Workaround: N/A
	Discovered in Release: 23.07-0.5.0.0
3464337	**Description:** Simultaneously adding or removing TC rules while operating on kernel version 6.3 could potentially result in stability issues.
	Keywords: ASAP, rules, TC
	Workaround: Make sure the following fix is part of the kernel: https://lore.kernel.org/netdev/20230504181616.2834983-3-vladbu@nvidia.com/T/
	Discovered in Release: 23.07-0.5.0.0
3469484	**Description:** Mirror and connection tracking (CT) offload actions are not supported simultaneously if the kernel version does not support hardware miss to TC actions. Thus, when performing a CT offload test, the actual number of offloaded connections may be lower than expected.
	Keywords: ASAP, CT offload
	Workaround: Make sure to have the following offending commit in the tree: net/sched: act_ct: offload UDP NEW connections Make sure to to have https://www.spinics.net/lists/stable-commits/msg303536.html in the kernel tree to fix this issue.
	Discovered in Release: 23.07-0.5.0.0
3473331	**Description:** When performing a CT offload test, the actual number of offloaded connections may be lower than expected.
	Keywords: ASAP, CT offload
	Workaround: N/A
	Discovered in Release: 23.07-0.5.0.0
3499413	**Description:** Due to the following kernel issue, under heavy load, some connections may not be offloaded, leading to performance issues: "net/sched: act_ct: offload UDP NEW connections."
	Keywords: ASAP, CT offload
	Workaround: N/A
	Discovered in Release: 23.07-0.5.0.0

Internal Ref. Number	Issue
3360710	**Description:** Configuring PFC in parallel to buffer size and prio2buffer commands may lead to misalignment between firmware and software in regards to receiving buffer ownership.
	Keywords: NetDev, PFC, Buffer Size, prio2buffer
	Workaround: First, configure PFC on all ports, and then perform other needed QoS (i.e., buffer_size or prio2buffer) configurations accordingly.
	Discovered in Release: 23.04-0.5.3.3
3413879	**Description:** OpenSM may not be started automatically if chkconfig was not installed before OpenSM is installed. Note, however, that chkconfig will fail to install if the directory (rather than symbolic link to directory) /etc/init.d already exists (e.g., from a previous installation of MLNX_OFED).
	Keywords: Installation, OpenSM, chkconfig
	Workaround: Install chkconfig before installing MLNX_OFED. If installing it fails, make sure /etc/init.d does not exist at the time of installing it.
	Discovered in Release: 23.04-0.5.3.3
3424596	**Description:** On SLES 15.4, installing MLNX_OFED using a package repository (with zypper) may trigger an error message about missing dependency for 'librte_eal.so.20.0()(64bit)' . This is because the inbox package libdpdk-20_0 is being uninstalled as it is incompatible with the MLNX_OFED rdma-core packages.
	Keywords: Installation, SLES 15.4
	Workaround: Uninstall the relevant packages: 'zypper uninstall libdpdk-20_0' before installing MLNX_OFED. This will also remove the inbox openvswitch package.
	Discovered in Release: 23.04-0.5.3.3
3433416	**Description:** On systems that were installed with MLNX_OFED 5.9 or older and include a CUDA package (ucx-cuda / hcoll-cuda), an upgrade to MLNX_OFED 23.04 using the package manager ("yum") method will fail. This is because MLNX_OFED up to 5.9 is built with CUDA 11. MLNX_OFED 23.04 is built with CUDA 12 and those CUDA versions are incompatible.
	Keywords: Installation, CUDA, yum
	Workaround: Remove CUDA packages included with OFED (ucx-cuda, hcoll-cuda) before upgrading. This will allow to upgrade MLNX_OFED regardless of CUDA version installed. To install them later, CUDA 12 must be installed on the system.
	Discovered in Release: 23.04-0.5.3.3
3420831	**Description:** mlx-steering-dump is not supported on systems in which Python3 is not the default.
	Keywords: mlx-steering-dump, Python3
	Workaround: N/A
	Discovered in Release: 23.04-0.5.3.3
3351989	**Description:** If the underlying persistent device name exceeds 15 characters in length, the operating system will not be able to perform renaming (i.e., the device name will remain "eth<digit>").
	Keywords: Persistant Interface Names
	Workaround: Add the --copy-ifnames-udev flag to the OFED installation command. Note that this flag is only applicable if the persistent name provided by the kernel, without the 'np<digit>' suffix, is 15 characters or fewer.

Internal Ref. Number	Issue
	Discovered in Release: 23.04-0.5.3.3

3 User Manual

3.1 Introduction

This manual is intended for system administrators responsible for the installation, configuration, management and maintenance of the software and hardware of VPI (InfiniBand, Ethernet) adapter cards. It is also intended for application developers.

NVIDIA OFED is a single Virtual Protocol Interconnect (VPI) software stack which operates across all NVIDIA network adapter solutions supporting the following uplinks to servers:

Uplink/Adapter Card	Driver Name	Uplink Speed
BlueField-2	mlx5	• InfiniBand: SDR, FDR, EDR, HDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE[2], 100GbE[2]
BlueField		• InfiniBand: SDR, QDR, FDR, FDR10, EDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE, 100GbE
ConnectX-7		• InfiniBand: EDR, HDR100, HDR, NDR200, NDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE[2], 100GbE[2], 200GbE[3]
ConnectX-6 Lx		• Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE[2]
ConnectX-6 Dx		• Ethernet: 10GbE, 25GbE, 40GbE, 50GbE[2], 100GbE[2], 200GbE[2]
ConnectX-6		• InfiniBand: SDR, FDR, EDR, HDR • Ethernet: 10GbE, 25GbE, 40GbE, 50GbE[2], 100GbE[2], 200GbE[2]
ConnectX-5/ConnectX-5 Ex		• InfiniBand: SDR, QDR, FDR, FDR10, EDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE, 100GbE
ConnectX-4 Lx		• Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE
ConnectX-4		• InfiniBand: SDR, QDR, FDR, FDR10, EDR • Ethernet: 1GbE, 10GbE, 25GbE, 40GbE, 50GbE, 56GbE[1], 100GbE

1. 56GbE is an NVIDIA proprietary link speed and can be achieved while connecting an NVIDIA adapter card to NVIDIA SX10XX switch series or when connecting an NVIDIA adapter card to another NVIDIA adapter card.
2. Speed that supports both NRZ and PAM4 modes in Force mode and Auto-Negotiation mode.
3. Speed that supports PAM4 mode only.

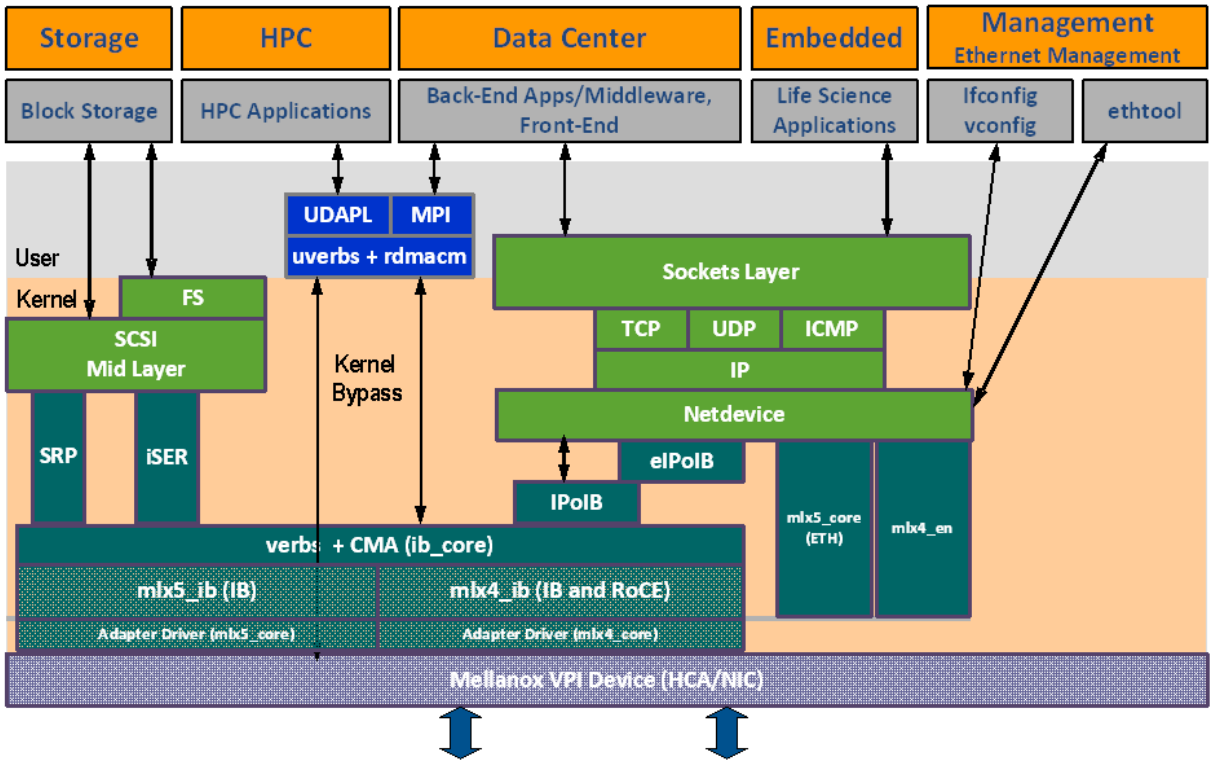
All NVIDIA network adapter cards are compatible with OpenFabrics-based RDMA protocols and software and are supported by major operating system distributions.

NVIDIA OFED is certified with the following products:

- NVIDIA Messaging Accelerator (VMA™) software: Socket acceleration library that performs OS bypass for standard socket-based applications.
  Please note, VMA support is provided separately from NVIDIA OFED support. For further information, please refer to the VMA documentation (docs.nvidia.com/networking/category/vma).
- NVIDIA Unified Fabric Manager (UFM®) software: Powerful platform for managing demanding scale-out computing fabric environments, built on top of the OpenSM industry standard routing engine.
- Fabric Collective Accelerator (FCA)—FCA is a NVIDIA MPI-integrated software package that utilizes CORE-Direct technology for implementing the MPI collectives communications.

3.1.1 Stack Architecture

The figure below shows a diagram of the NVIDIA OFED stack, and how upper layer protocols (ULPs) interface with the hardware and with the kernel and userspace. The application level also shows the versatility of markets that NVIDIA OFED applies to.



The following subsections briefly describe the various components of the NVIDIA OFED stack.

mlx4 VPI Driver

⚠ This driver is no longer supported in MLNX_OFED. To work with ConnectX-3® and ConnectX-3 Pro NICs, please refer to MLNX_OFED LTS version available on the web.

mlx5 Driver

mlx5 is the low-level driver implementation for the Connect-IB® and ConnectX-4 and above adapters designed by NVIDIA. ConnectX-4 and above adapter cards operate as a VPI adapter (Infiniband and Ethernet). The mlx5 driver is comprised of the following kernel modules:

⚠ Please note that Connect-IB card is no longer supported in MLNX_OFED. To work with this card, please refer to MLNX_OFED LTS version available on the web.

mlx5_core

Acts as a library of common functions (e.g. initializing the device after reset) required by ConnectX-4 and above adapter cards. mlx5_core driver also implements the Ethernet interfaces for ConnectX-4 and above. mlx5 drivers do not require the mlx5_en module as the Ethernet functionalities are built-in in the mlx5_core module.
mlx5_ib
Handles InfiniBand-specific functions and plugs into the InfiniBand mid layer.

libmlx5

libmlx5 is the provider library that implements hardware specific user-space functionality. If there is no compatibility between the firmware and the driver, the driver will not load and a message will be printed in the dmesg.
The following are the libmlx5 Legacy and RDMA-Core environment variables:

- MLX5_FREEZE_ON_ERROR_CQE
  - Causes the process to hang in a loop of completion with error, which is not flushed with error or retry exceeded occurs/
  - Otherwise disabled
- MLX5_POST_SEND_PREFER_BF
  - Configures every work request that can use blue flame will use blue flame
- Otherwise, blue flame depends on the size of the message and inline indication in the packet
- MLX5_SHUT_UP_BF
  - Disables blue flame feature
  - Otherwise, do not disable
- MLX5_SINGLE_THREADED
  - All spinlocks are disabled
  - Otherwise, spinlocks enabled
  - Used by applications that are single threaded and would like to save the overhead of taking spinlocks.
- MLX5_CQE_SIZE
  - 64—completion queue entry size is 64 bytes (default)
  - 128—completion queue entry size is 128 bytes
- MLX5_SCATTER_TO_CQE
  - Small buffers are scattered to the completion queue entry and manipulated by the driver. Valid for RC transport.

- Default is 1, otherwise disabled

The following are libmlx5 Legacy only environment variables:

- MLX5_ENABLE_CQE_COMPRESSION
  - Saves PCIe bandwidth by compressing a few CQEs into a smaller amount of bytes on PCIe. Setting this variable enables CQE compression.
  - Default value 0 (disabled)
- MLX5_RELAXED_PACKET_ORDERING_ON
  See "Out-of-Order (OOO) Data Placement" section.

Mid-layer Core

Core services include management interface (MAD), connection manager (CM) interface, 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, CM and management.

Upper Layer Protocols (ULPs)

IP over IB (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), in case RC is supported. The interface supports unicast, multicast and broadcast. For details, see "IP over InfiniBand (IPoIB)" section.

iSCSI Extensions for RDMA (iSER)

iSCSI Extensions for RDMA (iSER) extends the iSCSI protocol to RDMA. It permits data to be transferred directly into and out of SCSI buffers without intermediate data copies. For further information, please refer to "iSCSI Extensions for RDMA (iSER)" section.

SCSI RDMA Protocol (SRP)

SCSI RDMA Protocol (SRP) is designed to take full advantage of the protocol offload and RDMA features provided by the InfiniBand architecture. SRP allows a large body of SCSI software to be readily used on InfiniBand architecture. The SRP driver—known as the SRP Initiator—differs from traditional low-level SCSI drivers in Linux. The SRP Initiator does not control a local HBA; instead, it controls a connection to an I/O controller—known as the SRP Target—to provide access to remote storage devices across an InfiniBand fabric. The SRP Target resides in an I/O unit and provides storage services. See "SRP—SCSI RDMA Protocol" section.

User Direct Access Programming Library (uDAPL)

User Direct Access Programming Library (uDAPL) is a standard API that promotes data center application data messaging performance, scalability, and reliability over RDMA interconnects InfiniBand and RoCE. The uDAPL interface is defined by the DAT collaborative. This release of the uDAPL reference implementation package for both DAT 1.2 and 2.0 specification is timed to coincide with OFED release of the Open Fabrics (openfabrics.org) software stack.

MPI

Message Passing Interface (MPI) is a library specification that enables the development of parallel software libraries to utilize parallel computers, clusters, and heterogeneous networks. NVIDIA OFED includes the following MPI implementation over InfiniBand:

- Open MPI – an open source MPI-2 implementation by the Open MPI Project

NVIDIA OFED also includes MPI benchmark tests such as OSU BW/LAT, Intel MPI BeBenchmarkand Presta.

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 installed as part of NVIDIA OFED[1].

1. OpenSM is disabled by default. See "NVIDIA SM" section for details on enabling it.

Diagnostic Utilities

NVIDIA OFED includes the following two diagnostic packages for use by network and data center managers:

- ibutils—NVIDIA diagnostic utilities
- infiniband-diags—OpenFabrics Alliance InfiniBand diagnostic tools

NVIDIA Firmware Tools (MFT)

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

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

MFT includes a set of tools used for performing firmware update and configuration, as well as debug and diagnostics, and provides MST service. For the full list of available tools within MFT, please refer to MFT documentation (docs.nvidia.com/networking/category/mft).

3.1.2 Package Contents

3.1.2.1 ISO Image

NVIDIA OFED for Linux (MLNX_OFED_LINUX) is provided as ISO images or as a tarball, one per supported Linux distribution and CPU architecture, that includes *source code* and *binary* RPMs, firmware, utilities, and documentation. The ISO image contains an installation script (called mlnxofedinstall) that performs the necessary steps to accomplish the following:

- Discover the currently installed kernel
- Uninstall any InfiniBand stacks that are part of the standard operating system distribution or another vendor's commercial stack
- Install the MLNX_OFED_LINUX binary RPMs (if they are available for the current kernel)
- Identify the currently installed InfiniBand HCAs and perform the required firmware updates

3.1.2.2 Software Components

MLNX_OFED_LINUX contains the following software components:

- NVIDIA Host Channel Adapter Drivers
    - mlx5
        - mlx5_ib
        - mlx5_core (includes Ethernet)
    - Mid-layer core
        - Verbs, MADs, SA, CM, CMA, uVerbs, uMADs
    - Upper Layer Protocols (ULPs)
        - IPoIB, SRP Initiator and SRP
    - MPI
        - Open MPI stack supporting the InfiniBand, RoCE and Ethernet interfaces
        - MPI benchmark tests (OSU BW/LAT, Intel MPI Benchmark, Presta)
    - OpenSM: InfiniBand Subnet Manager
    - Utilities
        - Diagnostic tools
        - Performance tests
        - Sysinfo (see Sysinfo User Manual)
    - Firmware tools (MFT)
    - Source code for all the OFED software modules (for use under the conditions mentioned in the modules' LICENSE files)
    - Documentation

3.1.2.3 Firmware

The ISO image includes the following firmware item:

- mlnx-fw-updater RPM/DEB package, which contains firmware binaries for supported devices (using mlxfwmanager tool).

3.1.2.4 Directory Structure

The ISO image of MLNX_OFED_LINUX contains the following files and directories:

- mlnxofedinstall—the MLNX_OFED_LINUX installation script.
- ofed_uninstall.sh—This is the MLNX_OFED_LINUX un-installation script.
- <RPMS folders>—Directory of binary RPMs for a specific CPU architecture.
- src/—Directory of the OFED source tarball.

> ⚠ MLNX_OFED includes the OFED source RPM packages used as a build platform for kernel code but does not include the sources of NVIDIA proprietary packages.

- mlnx_add_kernel_support.sh—Script required to rebuild MLNX_OFED_LINUX for customized kernel version on supported Linux Distribution
- RPM based—A script required to rebuild MLNX_OFED_LINUX for customized kernel version on supported RPM-based Linux Distribution
- docs/—Directory of NVIDIA OFED related documentation

3.1.3 Module Parameters

mlx5_core Module Parameters

The mlx5_core module supports a single parameter used to select the profile which defines the number of resources supported.

prof_sel	The parameter name for selecting the profile. The supported values for profiles are: • 0—for medium resources, medium performance • 1—for low resources • 2—for high performance (int) (default)
guids	charp
node_guid	guids configuration. This module parameter will be obsolete!
debug_mask	debug_mask: 1 = dump cmd data, 2 = dump cmd exec time, 3 = both. Default=0 (uint)
probe_vf	probe VFs or not, 0 = not probe, 1 = probe. Default = 1 (bool)
num_of_groups	Controls the number of large groups in the FDB flow table. Default=4; Range=1-1024

ib_core Parameters

send_queue_size	Size of send queue in number of work requests (int)
recv_queue_size	Size of receive queue in number of work requests (int)
force_mr	Force usage of MRs for RDMA READ/WRITE operations (bool)
roce_v1_noncompat_gid	Default GID auto configuration (Default: yes) (bool)

ib_ipoib Parameters

max_nonsrq_conn_qp	Max number of connected-mode QPs per interface (applied only if shared receive queue is not available) (int)
mcast_debug_level	Enable multicast debug tracing if > 0 (int)
send_queue_size	Number of descriptors in send queue (int)
recv_queue_size	Number of descriptors in receive queue (int)
debug_level	Enable debug tracing if > 0 (int)
ipoib_enhanced	Enable IPoIB enhanced for capable devices (default = 1) (0-1) (int)

3.1.4 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.

3.2 Installation

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

The chapter contains the following sections:

3.2.1 Hardware and Software Requirements

Requirements	Description
Platforms	A server platform with an adapter card based on one of the following NVIDIA' VPI HCA devices listed in Supported NICs Speeds table. For the list of supported architecture platforms, please refer to the NVIDIA OFED Release Notes.
Required Disk Space for Installation	1GB
Device ID	For the latest list of device IDs, please visit NVIDIA website.
Operating System	Linux operating system. For the list of supported operating system distributions and kernels, please refer to the NVIDIA OFED Release Notes.
Installer Privileges	The installation requires administrator (root) privileges on the target machine.

3.2.2 Downloading the Drivers

1. Verify that the system has a NVIDIA network adapter (HCA/NIC) installed.
   The following example shows a system with an installed NVIDIA HCA:

```
# lspci -v | grep Mellanox
86:00.0 Network controller [0207]: Mellanox Technologies MT27620 Family
        Subsystem: Mellanox Technologies Device 0014
86:00.1 Network controller [0207]: Mellanox Technologies MT27620 Family
```

```
        Subsystem: Mellanox Technologies Device 0014
```

Note: For ConnectX-5 Socket Direct adapters, use ibdev2netdev to display the installed card and the mapping of logical ports to physical ports. Example:

```
[root@gen-l-vrt-203 ~]# ibdev2netdev -v | grep -i MCX556M-ECAT-S25
0000:84:00.0 mlx5_10 (MT4119 - MCX556M-ECAT-S25SN) CX556M - ConnectX-5 QSFP28 fw 16.22.0228 port 1 (DOWN )
==> p2p1 (Down)
0000:84:00.1 mlx5_11 (MT4119 - MCX556M-ECAT-S25SN) CX556M - ConnectX-5 QSFP28 fw 16.22.0228 port 1 (DOWN )
==> p2p2 (Down)
0000:05:00.0 mlx5_2 (MT4119 - MCX556M-ECAT-S25SN) CX556M - ConnectX-5 QSFP28 fw 16.22.0228 port 1 (DOWN )
==> p5p1 (Down)
0000:05:00.1 mlx5_3 (MT4119 - MCX556M-ECAT-S25SN) CX556M - ConnectX-5 QSFP28 fw 16.22.0228 port 1 (DOWN )
==> p5p2 (Down)
```

- Each PCI card of ConnectX-5 Socket Direct has a different PCI address. In the output example above, the first two rows indicate that one card is installed in a PCI slot with PCI Bus address 84 (hexadecimal), and PCI Device Number 00, and PCI Function Number 0 and 1. RoCE assigned mlx5_10 as the logical port, which is the same as netdevice p2p1, and both are mapped to physical port of PCI function 0000:84:00.0.
- RoCE logical port mlx5_2 of the second PCI card (PCI Bus address 05) and netdevice p5p1 are mapped to physical port of PCI function 0000:05:00.0, which is the same physical port of PCI function 0000:84:00.0.
  MT4119 is the PCI Device ID of the ConnectX-5 adapters family.

For more details, please refer to ConnectX-5 Socket Direct Hardware User Manual, available at nvidia.com/en-us/networking/.

2. Download the ISO image to your host.
   The image's name has the format MLNX_OFED_LINUX-<ver>-<OS label><CPU arch>.iso. It can be download from nvidia.com/en-us/networking/ → Products → Software → InfiniBand Drivers → MLNX_OFED.
   a. Scroll down to the Download wizard, and click the Download tab.
   b. Choose your relevant package depending on your host operating system.
   c. Click the desired ISO/tgz package.
   d. To obtain the download link, accept the End User License Agreement (EULA).
3. Use the md5sum utility to confirm the file integrity of your ISO image. Run the following command and compare the result to the value provided on the download page.

```
host1$ md5sum MLNX_OFED_LINUX-<ver>-<OS label>.iso
```

3.2.3 Installing MLNX_OFED

For installation of MLNX_OFED on Community OSs, please see section "Installation on Community Operating Systems" below.

3.2.3.1 Installation Script

The installation script, mlnxofedinstall, performs the following:

- Discovers the currently installed kernel

- Uninstalls any software stacks that are part of the standard operating system distribution or another vendor's commercial stack
- Installs the MLNX_OFED_LINUX binary RPMs (if they are available for the current kernel)
- Identifies the currently installed InfiniBand and Ethernet network adapters and automatically upgrades the firmware
  Note: To perform a firmware upgrade using customized firmware binaries, a path can be provided to the folder that contains the firmware binary files, by running --fw-image-dir. Using this option, the firmware version embedded in the MLNX_OFED package will be ignored.
  Example:

```
./mlnxofedinstall --fw-image-dir /tmp/my_fw_bin_files
```

If the driver detects unsupported cards on the system, it will abort the installation procedure. To avoid this, make sure to add `--skip-unsupported-devices-check` flag during installation.

Usage

```
./mnt/mlnxofedinstall [OPTIONS]
```

The installation script removes all previously installed OFED packages and re-installs from scratch. You will be prompted to acknowledge the deletion of the old packages.

Pre-existing configuration files will be saved with the extension ".conf.rpmsave".

- If you need to install OFED on an entire (homogeneous) cluster, a common strategy is to mount the ISO image on one of the cluster nodes and then copy it to a shared file system such as NFS. To install on all the cluster nodes, use cluster-aware tools (suchaspdsh).
- If your kernel version does not match with any of the offered pre-built RPMs, you can add your kernel version by using the "mlnx_add_kernel_support.sh" script located inside the MLNX_OFED package.

  On Redhat and SLES distributions with errata kernel installed there is no need to use the mlnx_add_kernel_support.sh script. The regular installation can be performed and weak-updates mechanism will create symbolic links to the MLNX_OFED kernel modules.

  If you regenerate kernel modules for a custom kernel (using `--add-kernel-support`), the packages installation will not involve automatic regeneration of the initramfs. In some cases, such as a system with a root filesystem mounted over a ConnectX card, not regenerating the initramfs may even cause the system to fail to reboot.

  In such cases, the installer will recommend running the following command to update the initramfs:

```
dracut -f
```

On some OSs, `dracut -f` might result in the following error message which can be safely ignore.

`libkmod: kmod_module_new_from_path: kmod_module 'mdev' already exists with different path`

The “mlnx_add_kernel_support.sh” script can be executed directly from the mlnxofedinstall script. For further information, please see '--add-kernel-support' option below.

On Ubuntu and Debian distributions drivers installation use Dynamic Kernel Module Support (DKMS) framework. Thus, the drivers' compilation will take place on the host during MLNX_OFED installation.
Therefore, using "mlnx_add_kernel_support.sh" is irrelevant on Ubuntu and Debian distributions.

Example: The following command will create a MLNX_OFED_LINUX ISO image for RedHat 7.3 under the /tmp directory.

```
# ./MLNX_OFED_LINUX-x.x-x-rhel7.3-x86_64/mlnx_add_kernel_support.sh -m /tmp/MLNX_OFED_LINUX-x.x-x-
rhel7.3-x86_64/ --make-tgz
Note: This program will create MLNX_OFED_LINUX TGZ for rhel7.3 under /tmp directory.
All Mellanox, OEM, OFED, or Distribution IB packages will be removed.
Do you want to continue?[y/N]:y
See log file /tmp/mlnx_ofed_iso.21642.log

Building OFED RPMs. Please wait...
Removing OFED RPMs...
Created /tmp/MLNX_OFED_LINUX-x.x-x-rhel7.3-x86_64-ext.tgz
```

- The script adds the following lines to /etc/security/limits.conf for the userspace components such as MPI:
  - * soft memlock unlimited
  - * hard memlock unlimited
    - These settings set the amount of memory that can be pinned by a userspace application to unlimited. If desired, tune the value unlimited to a specific amount of RAM.

For your machine to be part of the InfiniBand/VPI fabric, a Subnet Manager must be running on one of the fabric nodes. At this point, OFED for Linux has already installed the OpenSM Subnet Manager on your machine.

For the list of installation options, run: ./mlnxofedinstall --h

3.2.3.2 Installation Procedure

This section describes the installation procedure of MLNX_OFED on NVIDIA adapter cards.

1. Log in to the installation machine as root.
2. Mount the ISO image on your machine.

```
host1# mount -o ro,loop MLNX_OFED_LINUX-<ver>-<OS label>-<CPU arch>.iso /mnt
```

3. Run the installation script.

```
/mnt/mlnxofedinstall
Logs dir: /tmp/MLNX_OFED_LINUX-x.x-x.logs
This program will install the MLNX_OFED_LINUX package on your machine.
Note that all other Mellanox, OEM, OFED, RDMA or Distribution IB packages will be removed.
Those packages are removed due to conflicts with MLNX_OFED_LINUX, do not reinstall them.
Starting MLNX_OFED_LINUX-x.x.x installation ...
........
........
Installation finished successfully.
Attempting to perform Firmware update...
Querying Mellanox devices firmware ...
```

For unattended installation, use the --force installation option while running the MLNX_OFED installation script:

`/mnt/mlnxofedinstall --force`

MLNX_OFED for Ubuntu should be installed with the following flags in chroot environment:
./mlnxofedinstall --without-dkms --add-kernel-support --kernel <kernel version in chroot> --without-fw-update --force
For example:
./mlnxofedinstall --without-dkms --add-kernel-support --kernel 3.13.0-85-generic --without-fw-update --force
Note that the path to kernel sources (--kernel-sources) should be added if the sources are not in their default location.

In case your machine has the latest firmware, no firmware update will occur and the installation script will print at the end of installation a message similar to the following:
Device #1:

Device Type: ConnectX4
Part Number: MCX456A-ECA
Description: ConnectX-4 VPI adapter card; EDR IB (100Gb/s) and 100GbE; dual-port QSFP28; PCIe3.0 x16; ROHS R6
PSID: MT_2190110032
PCI Device Name: 0b:00.0
Base MAC: 0000e41d2d5cf810
Versions: Current Available
FW 12.14.0114 12.14.0114
Status: Up to date

In case your machine has an unsupported network adapter device, no firmware update will occur and one of the error messages below will be printed. Please contact your hardware vendor for help with firmware updates.

Error message #1:
Device #1:

Device Type: ConnectX4
Part Number: MCX456A-ECA
Description: ConnectX-4 VPI adapter card; EDR IB (100Gb/s) and 100GbE; dual-port QSFP28; PCIe3.0 x16; ROHS R6
PSID: MT_2190110032
PCI Device Name: 0b:00.0
Base MAC: 0000e41d2d5cf810
Versions: Current Available
FW 12.14.0114 N/A
Status: No matching image found

Error message #2:
The firmware for this device is not distributed inside NVIDIA driver: 0000:01:00.0 (PSID: IBM2150110033)
To obtain firmware for this device, please contact your HW vendor.

4. Case A: If the installation script has performed a firmware update on your network adapter, you need to either restart the driver or reboot your system before the firmware update can take effect. Refer to the table below to find the appropriate action for your specific card.

Action \ Adapter	Driver Restart	Standard Reboot (Soft Reset)	Cold Reboot (Hard Reset)
Standard ConnectX-4/ ConnectX-4 Lx or higher	-	+	-
Adapters with Multi-Host Support	-	-	+
Socket Direct Cards	-	-	+

   Case B: If the installations script has not performed a firmware upgrade on your network adapter, restart the driver by running: “/etc/init.d/openibd restart”.

5. (InfiniBand only) Run the hca_self_test.ofed utility to verify whether or not the InfiniBand link is up. The utility also checks for and displays additional information such as:
   - HCA firmware version
   - Kernel architecture
   - Driver version
   - Number of active HCA ports along with their states
   - Node GUID

     For more details on hca_self_test.ofed, see the file docs/readme_and_user_manual/hca_self_test.readme.

After installation completion, information about the OFED installation, such as prefix, kernel version, and installation parameters can be retrieved by running the command /etc/infiniband/info. Most of the OFED components can be configured or reconfigured after the installation, by modifying the relevant configuration files. See the relevant chapters in this manual for details.

The list of the modules that will be loaded automatically upon boot can be found in the /etc/infiniband/openib.conf file.

> ⚠ Installing OFED will replace the RDMA stack and remove existing 3rd party RDMA connectors.

3.2.3.3 Installation Results

Software	• Most of MLNX_OFED packages are installed under the "/usr" directory except for the following packages which are installed under the "/opt" directory:
	  ◦ fca and ibutils
	  ◦ iproute2 (rdma tool) - installed under /opt/Mellanox/iproute2/sbin/rdma
	• The kernel modules are installed under
	  ◦ /lib/modules/`uname -r`/updates on SLES and Fedora Distributions
	  ◦ /lib/modules/`uname -r`/extra/mlnx-ofa_kernel on RHEL and other RedHat like Distributions
	  ◦ /lib/modules/`uname -r`/updates/dkms/ on Ubuntu

Firmware	• The firmware of existing network adapter devices will be updated if the following two conditions are fulfilled:
	  ◦ The installation script is run in default mode; that is, without the option '--without- fw-update'
	  ◦ The firmware version of the adapter device is older than the firmware version included with the OFED ISO image
	  Note: If an adapter's Flash was originally programmed with an Expansion ROM image, the automatic firmware update will also burn an Expansion ROM image.
	• In case your machine has an unsupported network adapter device, no firmware update will occur and the error message below will be printed.
	"The firmware for this device is not distributed inside NVIDIA driver: 0000:01:00.0 (PSID: IBM2150110033)
	To obtain firmware for this device, please contact your HW vendor."

3.2.3.4 Installation Logging

While installing MLNX_OFED, the install log for each selected package will be saved in a separate log file.

The path to the directory containing the log files will be displayed after running the installation script in the following format:

Example:

```
Logs dir: /tmp/MLNX_OFED_LINUX-4.4-1.0.0.0.IBMM2150110033.logs
```

3.2.3.5 Driver Load Upon System Boot

Upon system boot, the NVIDIA drivers will be loaded automatically.

➢ *To prevent the automatic load of the NVIDIA drivers upon system boot:*

1. Add the following lines to the "/etc/modprobe.d/mlnx.conf" file.

```
blacklist mlx5_core
blacklist mlx5_ib
```

2. Set “ONBOOT=no” in the "/etc/infiniband/openib.conf" file.
3. If the modules exist in the initramfs file, they can automatically be loaded by the kernel. To prevent this behavior, update the initramfs using the operating systems’ standard tools.
   Note: The process of updating the initramfs will add the blacklists from step 1, and will prevent the kernel from loading the modules automatically.

3.2.3.6 mlnxofedinstall Return Codes

The table below lists the mlnxofedinstall script return codes and their meanings.

Return Code	Meaning
0	The Installation ended successfully
1	The installation failed
2	No firmware was found for the adapter device
22	Invalid parameter
28	Not enough free space
171	Not applicable to this system configuration. This can occur when the required hardware is not present on the system
172	Prerequisites are not met. For example, missing the required software installed or the hardware is not configured correctly
173	Failed to start the mst driver

Software	• Most of MLNX_OFED packages are installed under the “/usr” directory except for the following packages which are installed under the “/opt” directory:
	  ◦ fca and ibutils
	  ◦ iproute2 (rdma tool) - installed under /opt/Mellanox/iproute2/sbin/rdma
	• The kernel modules are installed under
	  ◦ /lib/modules/`uname -r`/updates on SLES and Fedora Distributions
	  ◦ /lib/modules/`uname -r`/extra/mlnx-ofa_kernel on RHEL and other RedHat like Distributions
	  ◦ /lib/modules/`uname -r`/updates/dkms/ on Ubuntu

Firmware	• The firmware of existing network adapter devices will be updated if the following two conditions are fulfilled:
	  ◦ The installation script is run in default mode; that is, without the option ‘--without- fw-update’
	  ◦ The firmware version of the adapter device is older than the firmware version included with the OFED ISO image
	  Note: If an adapter’s Flash was originally programmed with an Expansion ROM image, the automatic firmware update will also burn an Expansion ROM image.
	• In case your machine has an unsupported network adapter device, no firmware update will occur and the error message below will be printed.
	"The firmware for this device is not distributed inside NVIDIA driver: 0000:01:00.0 (PSID: IBM2150110033)
	To obtain firmware for this device, please contact your HW vendor."

3.2.3.7 Installation Logging

While installing MLNX_OFED, the install log for each selected package will be saved in a separate log file.

The path to the directory containing the log files will be displayed after running the installation script in the following format:

Example:

```
Logs dir: /tmp/MLNX_OFED_LINUX-4.4-1.0.0.0.IBMM2150110033.logs
```

3.2.3.8 Driver Load Upon System Boot

Upon system boot, the NVIDIA drivers will be loaded automatically.

To prevent the automatic load of the NVIDIA drivers upon system boot:

1. Add the following lines to the "/etc/modprobe.d/mlnx.conf" file.

   ```
   blacklist mlx5_core
   blacklist mlx5_ib
   ```

2. Set “ONBOOT=no” in the "/etc/infiniband/openib.conf" file.
3. If the modules exist in the initramfs file, they can automatically be loaded by the kernel. To prevent this behavior, update the initramfs using the operating systems’ standard tools.
   Note: The process of updating the initramfs will add the blacklists from step 1, and will prevent the kernel from loading the modules automatically.

3.2.3.9 mlnxofedinstall Return Codes

The table below lists the mlnxofedinstall script return codes and their meanings.

3.2.3.10 Installation on Community Operating Systems

NVIDIA provides OFED packages to be installed on common operating systems. These packages are provided as binaries, and NVIDIA provided full support for them. This model is now referred to as "Primary support".

Starting OFED 5.6, NVIDIA is introducing a new support model for OFED used on open source community operating systems. The goal of this new support model is to enable customers to use community-maintained variants of the Linux operating system, without being limited to major distributions that NVIDIA provides primary support for.

In the community model, there is shared responsibility between NVIDIA and customers choosing to use community operating systems in their environment. NVIDIA owns basic validation for the operating systems, so that customers know they can expect OFED to work. Customers are responsible for building their own packages and binaries (based on source code and build

instructions detailed below), and can also choose to deploy parts of OFED instead of the whole package.

In case of issues, for customers that are entitled for NVIDIA support (e.g. customers who have an applicable support contract), NVIDIA will do the best effort to assist, but may require the customer to work with the community to fix issues that are deemed to be caused by the community breaking OFED, as opposed to NVIDIA owning the fix end to end.

Overall, the following should be noted when running OFED on the community-supported operating systems:

- NVIDIA will perform sanity testing of OFED on community supported OSs
- NVIDIA will declare which kernel versions were tested (min/max), based on mainstream kernel (kernel.org) versions
- NVIDIA will not ship binaries/installation packages for community-supported OSs
- Customers will use source code of OFED and build guidance and will need to build their own binaries & installation packages
- Customers will be able to pick and choose the parts of OFED they deploy (e.g., drivers only, user tools only, and so forth)
- In case of bug reports, customers may be asked to reproduce on a primary-supported distribution (such as RHEL) in order to have support
- If issues are deemed specific to the community OS (e.g., if an OS deviates from mainstream in a way that breaks OFED), it is the customer's responsibility to work with the community to fix it

Below are the instructions how to build OFED from the sources provided by NVIDIA.

3.2.3.10.1 Installing OFED on Community Operating Systems

1. Download sources from https://network.nvidia.com/products/infiniband-drivers/linux/mlnx_ofed
    a. Scroll down to the and click the "Download" tab.
    b. Choose the relevant package, depending on the host operating system (the package format is what really matters (rpm/deb)).
    c. Download the desired tgz SOURCE package (e.g., MLNX_OFED_SRC-<debian?>-<ver>.tgz)

2. Unpack the tarball.

```
tar -xzf MLNX_OFED_SRC-<debian?>-<version>.tgz
```

3. Go to the extracted directory.

```
cd MLNX_OFED_SOURCE-5.6-x.x.x.x
```

4. Run the installation script with the relevant options.

```
./install.pl <option 1> <option 2> .
```

Use "`./install.pl --help`" to see all the options and choose the desired option. For example: "`./install.pl --all`" should build and install all the default packages.

5. While running `install.pl`, the script may fail on missing dependencies. Those dependencies should be installed manually before running `install.pl` again.

3.2.3.10.2 Proprietary Packages

The installation procedure does not install proprietary packages—propriety packages should be installed upon request.

List of closed source proprietary packages:

- Clusterkit
- DPCP
- hcoll
- Sharp
- ibutils2
- opensm

Today the only way to install these packages is by using an already-built rpm/deb from a similar primary operating system.

The following table maps the community OSs that are most similar to Primary OSs based on internal testing:

Community OS	Most Similar Primary OS
BCLINUX7.6	RHEL 7.9
BCLINUX7.7	RHEL 7.9
BCLINUX8.1	RHEL 8.5
Debian9.13	Debian 10.8
EulerOS2.0sp5	EulerOS2.0sp10
EulerOS2.0sp8	EulerOS2.0sp10
RHEL/CentOS7.5	RHEL 7.9
RHEL/CentOS7.5alternate	RHEL 7.9
RHEL/CentOS7.6alternate	RHEL 7.9
RHEL/CentOS7.7	RHEL 7.9
RHEL/CentOS7.8	RHEL 7.9
SLES12SP2	SLES12 SP5
SLES12SP3	SLES12 SP5
SLES12SP4	SLES12 SP5
Ubuntu16.04	Ubuntu22.04
Alma 8.5	RHEL 8.5
Anolis OS 8.4	RHEL 8.5
CentOS Stream 8.5	RHEL 8.5
Fedora 35	RHEL 8.5

Community OS	Most Similar Primary OS
OpenEuler OS 22.03 LTS	OpenEuler20 SP3
OpenSUSE 15.3	SLES15 SP3
Photon OS 3.0	RHEL 7.9
Rocky 8.5	RHEL 8.5
Rocky 8.6	RHEL 8.6

3.2.3.10.3 Download and install MFT and Firmware

NVIDIA Firmware Tools (MFT) Download

- https://network.nvidia.com/products/adapter-software/firmware-tools

Firmware Downloads

- https://network.nvidia.com/support/firmware/firmware-downloads

3.2.3.10.4 Creating Metapackages and Installing OFED on Multiple Servers (RPM-Based Systems)

By building the packages using `install.pl`, the RPMs will be created under `/tmp/OFED-internal-5.6-x.x.x/RPMS/<OS>/<arch>/`

1. List the packages that will be installed by running the following:

```
/tmp/OFED-internal-5.6-x.x.x/install.pl--all -p
```

2. Create working directories.
   Example:

```
mkdir -p /tmp/OFED_topdir/SOURCES/<rpmname>-<version>
mkdir -p /tmp/OFED_topdir/SPECS
mkdir -p /tmp/OFED_topdir/BUILD
mkdir -p /tmp/OFED_topdir/SRPMS
mkdir -p /tmp/OFED_topdir/RPMS
```

   For --all option it used to set *rpmname* as mlnx-ofed-all
   OFED will be set to the current GA version being used.
3. Create a new XML file under `/tmp/OFED_topdir/comp.xml` that contains all the required packages.
   Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<comps>
<group>

<id>all</id>
<name>MLNX_OFED ALL</name>
<default>true</default>
<description>Mellanox OpenFabrics Enterprise Distribution for Linux: MLNX_OFED ALL packages </description>
<uservisible>true</uservisible>
<packagelist>

/* ------ The list below needs to be aligned with the packages listed by "install.pl --all -p" ------ */

<packagereq type="default">mlnx-tools</packagereq>
<packagereq type="default">rdma-core-devel</packagereq>

<packagereq type="default">perftest</packagereq>
```

```
<packagereq type="default">Package Example 1</packagereq>

<packagereq type="default">Package Example 2</packagereq>
<packagereq type="default">mlnx-ofed-all</packagereq>  // This line is for the Group name
</packagelist>
</group>
</comps>
```

4. Create tarball with the correct source name:

```
echo <rpmname>-<version> > /tmp/OFED_topdir/SOURCES/<rpmname>-<version>/<rpmname>-release
cd /tmp/OFED_topdir/SOURCES
tar czf <rpmname>-<version>.tar.gz <rpmname>-<version>
```

5. Create new spec file that requires all the packages in that directory under `/tmp/OFED_topdir/SPECS/<rpmname>.spec`
6. Build the RPM metapackage from the spec file that was created before. As a result, the RPMs will be created under /tmp/OFED_topdir/RPMS/<arch>/

```
rpmbuild -ba --target noarch --define '_source_filedigest_algorithm md5' --define
'_binary_filedigest_algorithm md5' --define '_topdir /tmp/OFED_topdir' --define '_sourcedir %{_topdir}/
SOURCES' --define '_specdir %{_topdir}/SPECS' --define '_srcrpmdir %{_topdir}/SRPMS' --define '_rpmdir %
{_topdir}/RPMS' /tmp/OFED_topdir/SPECS/<rpmname>.spec
```

7. Copy the newly-created RPMs into the RPMS directory that was created before.

```
cp /tmp/OFED_topdir/RPMS/noarch/mlnx-ofed-all-5.6-.rhel8u3.noarch.rpm /tmp/OFED-internal-5.6-x.x.x/<arch>/
redhat-release-8.3-1.0.el8/x86_64/
```

8. Create repodata.

```
createrepo -q -g /tmp/OFED_topdir/comps.xml /tmp/OFED-internal-5.6-x.x.x/RPMS/redhat-release-8.3-1.0.el8/
<arch>/
```

9. Copy the generated RPMS to any server and do the following to install OFED:
    a. Create new yum repository.

```
cat /etc/yum.repos.d/mlnx_ofed.repo
[mlnx_ofed]
name=mlnx_ofed
baseurl=/tmp/OFED-internal-5.6-x.x.x/RPMS/OS/<arch>/
enabled=1
gpgcheck=0
```

    b. Refresh repository list.

```
yum repolist
```

    c. Install OFED.

```
yum install mlnx-ofed-all
```

3.2.3.10.5 Creating Metapackages and Installing OFED on Multiple Servers (Deb-Based Systems)

By building the packages using install.pl, the debs will be created under `/tmp/OFED-internal-5.6-x.x.x/DEBS/<OS>/<arch>/`

1. List the packages that will be installed by the following:

```
/tmp/OFED-internal-debian-5.6-x.x.x/install.pl --all -p
```

2. Create working directories.
   Example:

```
mkdir -p /tmp/OFED_topdir/<rpmname>-<version>/debian/source
echo <rpmname>-<version> > /tmp/OFED_topdir/<rpmname>-<version>/<rpmname>-release
```

   For `--all` option it used to set *rpmname* as mlnx-ofed-all
   For OFED 5.6, the *version* will be set to 5.6
3. Create rules file under `/tmp/OFED_topdir/<rpmname>-<version>/debian/rules`

```
#!/usr/bin/make -f
# -*- makefile -*-

export DH_OPTIONS
pname:=<rpmname>

%:
dh $@

override_dh_auto_install:
    dh_installdirs -p$(pname) usr/share/doc/$(pname)
    install -m 0644 $(pname)-release debian/$(pname)/usr/share/doc/$(pname)
```

4. Change rules file mode.

```
chmod 755 /tmp/OFED_topdir/<rpmname>-<version>/debian/rules
```

5. Create rules file under `/tmp/OFED_topdir/<rpmname>-<version>/debian/compat`

```
echo 9 > /tmp/OFED_topdir/<rpmname>-<version>/debian/compat
```

6. Create source/format file under `/tmp/OFED_topdir/<rpmname>-<version>/debian/source/format`

```
echo "3.0 (quilt)" > /tmp/OFED_topdir/<rpmname>-<version>/debian/source/format
```

7. Create changelog file under `/tmp/OFED_topdir/<rpmname>-<version>/debian/changelog`
   Exmaple:

```
mlnx-ofed-all (5.6-x.x.x.x) unstable; urgency=low

* Initial release

-- your username <mail> Sun, 15 May 2022 21:00:00 +0200
```

8. Create postinst script (if needed) under `/tmp/OFED_topdir/<rpmname>-<version>/debian/<rpmname>.postinst`

```
#!/bin/bash
cd /lib/modules
for dd in `/bin/ls`
do
/sbin/depmod $dd >/dev/null 2>&1
done

if [ -f /usr/bin/ofed_info ]; then
sed -i -r -e "s/^(OFED)(.*)(-[0-9]*.*-[0-9]*.*):/MLNX_OFED_LINUX-5.6-x.x.x.x (\1\3):\n/" /usr/bin/ofed_info
sed -i -r -e "s/(.*then echo) (.*):(.*)/\1 MLNX_OFED_LINUX-5.6-x.x.x.x: \3/" /usr/bin/ofed_info
sed -i -r -e "s/(.*X-n\" ]; then echo) (.*)(; exit.*)/\1 5.6-x.x.x.x \3/" /usr/bin/ofed_info
sed -i -e "s/OFED-internal/MLNX_OFED_LINUX/g" /usr/bin/ofed_info
```

```
fi

# Switch off opensmd service
/sbin/chkconfig --set opensmd off > /dev/null 2>&1 || true
/sbin/chkconfig opensmd off > /dev/null 2>&1 || true
if [ -f "/etc/init.d/opensmd" ] ; then
if [ -e /sbin/chkconfig ]; then
/sbin/chkconfig --del opensmd > /dev/null 2>&1 || true
elif [ -e /usr/sbin/update-rc.d ]; then
/usr/sbin/update-rc.d -f opensmd remove > /dev/null 2>&1 || true
else
/usr/lib/lsb/remove_initd /etc/init.d/opensmd > /dev/null 2>&1 || true
fi
fi

# Disable ibacm daemon by default
chkconfig --del ibacm > /dev/null 2>&1 || true

# disable SDP and QIB loading by default
if [ -e /etc/infiniband/openib.conf ]; then
sed -i -r -e "s/^SDP_LOAD=.*/SDP_LOAD=no/" /etc/infiniband/openib.conf
sed -i -r -e "s/^QIB_LOAD=.*/QIB_LOAD=no/" /etc/infiniband/openib.conf
fi

/sbin/ldconfig > /dev/null 2>&1 || true
```

9. Create control file under `/tmp/OFED_topdir/<rpmname>-<version>/debian/control`

```
Source: mlnx-ofed-all
Section: utils
Priority: extra
Maintainer: your username <mail>
Build-Depends: debhelper (>= 9.0.0)
Standards-Version: 3.9.2
Homepage: http://www.mellanox.com

Package:mlnx-ofed-all // PACKAGE NAME
Architecture: all

// list of the packages we need to install in this group (see the list of packages to install we created at
first)

Depends: ${shlibs:Depends}, ${misc:Depends}, dpcp (>=1.1.25-1.56076), mstflint (>=4.16.0-1.56076), rdmacm-
utils (>=56mlnx40-1.56076), mlnx-tools (>=5.2.0-0.56076), mlnx-iproute2 (>=5.16.0-1.56076), opensm
(>=5.11.0.MLNX20220418.fd3d650-0.1.56076), ibutils2 (>=2.1.1-0.148.MLNX20220418.g60b8156.56076), ofed-
scripts (>=5.6-OFED.5.6.0.7.6), dump-pr (>=1.0-5.11.0.MLNX20220418.g7e9d922.56076), mlnx-ethtool
(>=5.15-1.56076), perftest (>=4.5-0.14.gd962d8c.56076), libibverbs-dev (>=56mlnx40-1.56076), ucx
(>=1.13.0-1.56076), ibsim-doc (>=0.10-1.56076), srp-dkms (>=5.6-OFED.5.6.0.7.6.1), ibacm
(>=56mlnx40-1.56076), sharp (>=2.7.0.MLNX20220331.8d57397a-1.56076), libibmad5 (>=56mlnx40-1.56076), mlnx-
ofed-kernel-utils (>=5.6-OFED.5.6.0.7.6.1), srptools (>=56mlnx40-1.56076), ibsim (>=0.10-1.56076),
libibmad-dev (>=56mlnx40-1.56076), libibumad3 (>=56mlnx40-1.56076), ibverbs-utils (>=56mlnx40-1.56076),
rdma-core (>=56mlnx40-1.56076), libibverbs1 (>=56mlnx40-1.56076), isert-dkms (>=5.6-OFED.5.6.0.7.6.1),
libibumad-dev (>=56mlnx40-1.56076), iser-dkms (>=5.6-OFED.5.6.0.7.6.1), ibdump (>=6.0.0-1.56076),
libopensm-devel (>=5.11.0.MLNX20220418.fd3d650-0.1.56076), libopensm
(>=5.11.0.MLNX20220418.fd3d650-0.1.56076), kernel-mft-dkms (>=4.20.0-29), libibnetdisc5
(>=56mlnx40-1.56076), librdmacm1 (>=56mlnx40-1.56076), opensm-doc
(>=5.11.0.MLNX20220418.fd3d650-0.1.56076), mlnx-ofed-kernel-dkms (>=5.6-OFED.5.6.0.7.6.1), infiniband-diags
(>=56mlnx40-1.56076), mft (>=4.15.1-9), librdmacm-dev (>=56mlnx40-1.56076), ibverbs-providers
(>=56mlnx40-1.56076)
Description: MLNX_OFED all installer package (with DKMS support)

Package:mlnx-ofed-all-exact
Architecture: all
```

10. Create copyright file under `/tmp/OFED_topdir/<rpmname>-<version>/debian/copyright`
    On Debian systems, a copy of the General Public License version 2 could be found under `/usr/share/common-licenses/GPL-2`

```
Format: https://www.debian.org/doc/packaging-manuals/copyright-format/1.0

Files: *
Copyright: 2022, NVIDIA Corporation
License: GPLv2-and-2BSD
* This software is available to you under a choice of one of two
* licenses. You may choose to be licensed under the terms of the GNU
* General Public License (GPL) Version 2, available from the file
* COPYING in the main directory of this source tree, or the
* BSD license below:
*
* Redistribution and use in source and binary forms, with or
* without modification, are permitted provided that the following
* conditions are met:
*
* - Redistributions of source code must retain the above
* copyright notice, this list of conditions and the following
* disclaimer.
*
* - Redistributions in binary form must reproduce the above
* copyright notice, this list of conditions and the following
* disclaimer in the documentation and/or other materials
* provided with the distribution.
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
```

```
* EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
* MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
* NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
* BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
* ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
* CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
```

11. Creating tarball.

```
cd /tmp/OFED_topdir
tar czf <rpmname>_<version>.orig.tar.gz <rpmname>-<version>
cd <rpmname>-<version>
dpkg-buildpackage -us -uc
cp -af *.deb /tmp/OFED-internal-5.6-x.x.x/DEBS/OS/arch/
```

12. Create new yum repository under `/etc/yum.repos.d/mlnx_ofed.repo`

```
[mlnx_ofed]
deb [trusted=yes] file:/tmp/OFED-internal-5.6-0.7.6/DEBS/debian10.8/x86_64/ ./
```

13. Refresh repository list.

```
apt update
```

14. Install OFED.

```
apt install mlnx-ofed-all
```

mpitests package is disabled on Fedora 35 and Photon due to building issues.mpitests package is disabled on Fedora 35 and Photon due to building issues.

3.2.3.11 Additional Installation Procedures

3.2.3.11.1 Installing MLNX_OFED Using YUM

This type of installation is applicable to RedHat/OL and Fedora operating systems.

3.2.3.11.1.1 Setting up MLNX_OFED YUM Repository

1. Log into the installation machine as root.
2. Mount the ISO image on your machine and copy its content to a shared location in your network.

```
# mount -o ro,loop MLNX_OFED_LINUX-<ver>-<OS label>-<CPU arch>.iso /mnt
```

3. Download and install NVIDIA's GPG-KEY:
   The key can be downloaded via the following link:
   http://www.mellanox.com/downloads/ofed/RPM-GPG-KEY-Mellanox-SHA256

```
# wget http://www.mellanox.com/downloads/ofed/RPM-GPG-KEY-Mellanox-SHA256
--2018-01-25 13:52:30--  http://www.mellanox.com/downloads/ofed/RPM-GPG-KEY-Mellanox-SHA256
Resolving www.mellanox.com... 72.3.194.0
Connecting to www.mellanox.com|72.3.194.0|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1354 (1.3K) [text/plain]
Saving to: ?RPM-GPG-KEY-Mellanox-SHA256?
```

```
100%[=================================================>] 1,354       --.-K/s   in 0s

2018-01-25 13:52:30 (247 MB/s) - ?RPM-GPG-KEY-Mellanox-SHA256? saved [1354/1354]
```

4. Install the key.

```
# sudo rpm --import RPM-GPG-KEY-Mellanox-SHA256
warning: rpmts_HdrFromFdno: Header V3 DSA/SHA1 Signature, key ID 6224c050: NOKEY
Retrieving key from file:///repos/MLNX_OFED/<MLNX_OFED file>/RPM-GPG-KEY-Mellanox-SHA256
Importing GPG key 0x6224C050:
 Userid: "Mellanox Technologies (Mellanox Technologies - Signing Key v2) <support@mellanox.com>"
 From  : /repos/MLNX_OFED/<MLNX_OFED file>/RPM-GPG-KEY-Mellanox-SHA256
Is this ok [y/N]:
```

5. Check that the key was successfully imported.

```
# rpm -q gpg-pubkey --qf '%{NAME}-%{VERSION}-%{RELEASE}\t%{SUMMARY}\n' | grep Mellanox
gpg-pubkey-a9e4b643-520791ba    gpg(Mellanox Technologies <support@mellanox.com>)
```

6. Create a yum repository configuration file called "/etc/yum.repos.d/mlnx_ofed.repo" with the following content:

```
[mlnx_ofed]
name=MLNX_OFED Repository
baseurl=file:///<path to extracted MLNX_OFED package>/RPMS
enabled=1
gpgkey=file:///<path to the downloaded key RPM-GPG-KEY-Mellanox-SHA256>
gpgcheck=1
```

7. Check that the repository was successfully added.

```
# yum repolist
Loaded plugins: product-id, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to
register.
repo id repo name                                status
mlnx_ofed   MLNX_OFED Repository                         108
rpmforge    RHEL 6Server - RPMforge.net - dag            4,597

repolist: 8,351
```

Setting up MLNX_OFED YUM Repository Using --add-kernel-support

1. Log into the installation machine as root.
2. Mount the ISO image on your machine and copy its content to a shared location in your network.

```
# mount -o ro,loop MLNX_OFED_LINUX-<ver>-<OS label>-<CPU arch>.iso /mnt
```

3. Build the packages with kernel support and create the tarball.

```
# /mnt/mlnx_add_kernel_support.sh --make-tgz <optional --kmp> -k $(uname -r) -m /mnt/
Note: This program will create MLNX_OFED_LINUX TGZ for rhel7.6 under /tmp directory.
Do you want to continue?[y/N]:y
See log file /tmp/mlnx_iso.4120_logs/mlnx_ofed_iso.4120.log

Checking if all needed packages are installed...
Building MLNX_OFED_LINUX RPMS . Please wait...
Creating metadata-rpms for 3.10.0-957.21.3.el7.x86_64 ...
WARNING: If you are going to configure this package as a repository, then please note
WARNING: that it contains unsigned rpms, therefore, you need to disable the gpgcheck
WARNING: by setting 'gpgcheck=0' in the repository conf file.
Created /tmp/MLNX_OFED_LINUX-5.2-0.5.5.0-rhel7.6-x86_64-ext.tgz
```

4. Open the tarball.

```
# cd /tmp/
# tar -xvf /tmp/MLNX_OFED_LINUX-5.2-0.5.5.0-rhel7.6-x86_64-ext.tgz
```

5. Create a YUM repository configuration file called "/etc/yum.repos.d/mlnx_ofed.repo" with the following content:

```
[mlnx_ofed]
name=MLNX_OFED Repository
baseurl=file:///<path to extracted MLNX_OFED package>/RPMS
enabled=1
gpgcheck=0
```

6. Check that the repository was successfully added.

```
# yum repolist
Loaded plugins: product-id, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to
register.
repo id repo name                                status
mlnx_ofed   MLNX_OFED Repository                         108
rpmforge    RHEL 6Server - RPMforge.net - dag            4,597

repolist: 8,351
```

3.2.3.11.1.2 Installing MLNX_OFED Using the YUM Tool

After setting up the YUM repository for MLNX_OFED package, perform the following:

1. View the available package groups by invoking:

```
# yum search mlnx-ofed-
mlnx-ofed-all.noarch : MLNX_OFED all installer package  (with KMP support)
mlnx-ofed-all-user-only.noarch : MLNX_OFED all-user-only installer package  (User Space packages only)
mlnx-ofed-basic.noarch : MLNX_OFED basic installer package  (with KMP support)
mlnx-ofed-basic-user-only.noarch : MLNX_OFED basic-user-only installer package  (User Space packages only)
mlnx-ofed-bluefield.noarch : MLNX_OFED bluefield installer package  (with KMP support)
mlnx-ofed-bluefield-user-only.noarch : MLNX_OFED bluefield-user-only installer package  (User Space
packages only)
mlnx-ofed-dpdk.noarch : MLNX_OFED dpdk installer package  (with KMP support)
mlnx-ofed-dpdk-upstream-libs.noarch : MLNX_OFED dpdk-upstream-libs installer package  (with KMP support)
mlnx-ofed-dpdk-upstream-libs-user-only.noarch : MLNX_OFED dpdk-upstream-libs-user-only installer
package  (User Space packages only)
mlnx-ofed-dpdk-user-only.noarch : MLNX_OFED dpdk-user-only installer package  (User Space packages only)
mlnx-ofed-eth-only-user-only.noarch : MLNX_OFED eth-only-user-only installer package  (User Space packages
only)
mlnx-ofed-guest.noarch : MLNX_OFED guest installer package  (with KMP support)
mlnx-ofed-guest-user-only.noarch : MLNX_OFED guest-user-only installer package  (User Space packages only)
mlnx-ofed-hpc.noarch : MLNX_OFED hpc installer package  (with KMP support)
mlnx-ofed-hpc-user-only.noarch : MLNX_OFED hpc-user-only installer package  (User Space packages only)
mlnx-ofed-hypervisor.noarch : MLNX_OFED hypervisor installer package  (with KMP support)
mlnx-ofed-hypervisor-user-only.noarch : MLNX_OFED hypervisor-user-only installer package  (User Space
packages only)
mlnx-ofed-kernel-only.noarch : MLNX_OFED kernel-only installer package  (with KMP support)
mlnx-ofed-vma.noarch : MLNX_OFED vma installer package  (with KMP support)
mlnx-ofed-vma-eth.noarch : MLNX_OFED vma-eth installer package  (with KMP support)
mlnx-ofed-vma-eth-user-only.noarch : MLNX_OFED vma-eth-user-only installer package  (User Space packages
only)
mlnx-ofed-vma-user-only.noarch : MLNX_OFED vma-user-only installer package  (User Space packages only)
mlnx-ofed-vma-vpi.noarch : MLNX_OFED vma-vpi installer package  (with KMP support)
mlnx-ofed-vma-vpi-user-only.noarch : MLNX_OFED vma-vpi-user-only installer package  (User Space packages
only
```

where:

mlnx-ofed-all	Installs all available packages in MLNX_OFED
mlnx-ofed-basic	Installs basic packages required for running NVIDIA cards
mlnx-ofed-guest	Installs packages required by guest OS
mlnx-ofed-hpc	Installs packages required for HPC
mlnx-ofed-hypervisor	Installs packages required by hypervisor OS
mlnx-ofed-vma	Installs packages required by VMA
mlnx-ofed-vma-eth	Installs packages required by VMA to work over Ethernet
mlnx-ofed-vma-vpi	Installs packages required by VMA to support VPI
bluefield	Installs packages required for BlueField

dpdk	Installs packages required for DPDK
dpdk-upstream-libs	Installs packages required for DPDK using RDMA-Core
kernel-only	Installs packages required for a non-default kernel

Note: MLNX_OFED provides kernel module RPM packages with KMP support for RHEL and SLES. For other operating systems, kernel module RPM packages are provided only for the operating system's default kernel. In this case, the group RPM packages have the supported kernel version in their package's name.
Example:

```
mlnx-ofed-all-3.17.4-301.fc21.x86_64.noarch : MLNX_OFED all installer package for kernel
3.17.4-301.fc21.x86_64 (without KMP support)
mlnx-ofed-basic-3.17.4-301.fc21.x86_64.noarch : MLNX_OFED basic installer package for kernel
3.17.4-301.fc21.x86_64 (without KMP support)
mlnx-ofed-guest-3.17.4-301.fc21.x86_64.noarch : MLNX_OFED guest installer package for kernel
3.17.4-301.fc21.x86_64 (without KMP support)
mlnx-ofed-hpc-3.17.4-301.fc21.x86_64.noarch : MLNX_OFED hpc installer package for kernel
3.17.4-301.fc21.x86_64 (without KMP support)
mlnx-ofed-hypervisor-3.17.4-301.fc21.x86_64.noarch : MLNX_OFED hypervisor installer package for kernel
3.17.4-301.fc21.x86_64 (without KMP support)
mlnx-ofed-vma-3.17.4-301.fc21.x86_64.noarch : MLNX_OFED vma installer package for kernel
3.17.4-301.fc21.x86_64 (without KMP support)
mlnx-ofed-vma-eth-3.17.4-301.fc21.x86_64.noarch : MLNX_OFED vma-eth installer package for kernel
3.17.4-301.fc21.x86_64 (without KMP support)
mlnx-ofed-vma-vpi-3.17.4-301.fc21.x86_64.noarch : MLNX_OFED vma-vpi installer package for kernel
3.17.4-301.fc21.x86_64 (without KMP support)
mlnx-ofed-hypervisor-3.17.4-301.fc21.x86_64.noarch : MLNX_OFED hypervisor installer package for kernel
3.17.4-301.fc21.x86_64 (without KMP support)
mlnx-ofed-vma-3.17.4-301.fc21.x86_64.noarch : MLNX_OFED vma installer package for kernel
3.17.4-301.fc21.x86_64 (without KMP support)
mlnx-ofed-vma-eth-3.17.4-301.fc21.x86_64.noarch : MLNX_OFED vma-eth installer package for kernel
3.17.4-301.fc21.x86_64 (without KMP support)
mlnx-ofed-vma-vpi-3.17.4-301.fc21.x86_64.noarch : MLNX_OFED vma-vpi installer package for kernel
3.17.4-301.fc21.x86_64 (without KMP support)
```

When using an operating system different than RHEL or SLES, or you have installed a kernel that is not supported by default in MLNX_OFED, you can use the mlnx_add_kernel_support.sh script to build MLNX_OFED for your kernel.
The script will automatically build the matching group RPM packages for your kernel so that you can still install MLNX_OFED via yum.
Please note that the resulting MLNX_OFED repository will contain unsigned RPMs, therefore, you should set 'gpgcheck=0' in the repository configuration file.

2. Install the desired group.

```
# yum install mlnx-ofed-all
Loaded plugins: langpacks, product-id, subscription-manager
Resolving Dependencies
--> Running transaction check
---> Package mlnx-ofed-all.noarch 0:3.1-0.1.2 will be installed
--> Processing Dependency: kmod-isert = 1.0-OFED.3.1.0.1.2.1.g832a737.rhel7u1 for package: mlnx-ofed-
all-3.1-0.1.2.noarch
...................
...................
  qperf.x86_64 0:0.4.9-9
  rds-devel.x86_64 0:2.0.7-1.12
  rds-tools.x86_64 0:2.0.7-1.12
  sdpnetstat.x86_64 0:1.60-26
  srptools.x86_64 0:1.0.2-12

Complete!
```

Installing MLNX_OFED using the “YUM” tool does not automatically update the firmware. To update the firmware to the version included in MLNX_OFED package, run:

```
# yum install mlnx-fw-updater
```

OR:
Update the firmware to the latest version available on NVIDIA's website as described in "Updating Firmware After Installation" section.

3.2.3.11.2 Installing MLNX_OFED Using apt-get

This type of installation is applicable to Debian and Ubuntu operating systems.

3.2.3.11.2.1 Setting up MLNX_OFED apt-get Repository

1. Log into the installation machine as root.
2. Extract the MLNX_OFED package on a shared location in your network.
   It can be downloaded from https://www.nvidia.com/en-us/networking/ → Products → Software → InfiniBand Drivers.
3. Create an apt-get repository configuration file called "/etc/apt/sources.list.d/mlnx_ofed.list" with the following content:

```
deb file:/<path to extracted MLNX_OFED package>/DEBS ./
```

4. Download and install NVIDIA's Technologies GPG-KEY.

```
# wget -qO - http://www.mellanox.com/downloads/ofed/RPM-GPG-KEY-Mellanox-SHA256-SHA256 | sudo apt-key add -
```

5. Verify that the key was successfully imported.

```
# apt-key list
pub   1024D/A9E4B643 2013-08-11
uid   Mellanox Technologies <support@mellanox.com>
sub   1024g/09FCC269 2013-08-11
```

6. Update the apt-get cache.

```
# sudo apt-get update
```

Setting up MLNX_OFED apt-get Repository Using --add-kernel-support

1. Log into the installation machine as root.
2. Mount the ISO image on your machine and copy its content to a shared location in your network.

```
# mount -o ro,loop MLNX_OFED_LINUX-<ver>-<OS label>-<CPU arch>.iso /mnt
```

3. Build the packages with kernel support and create the tarball.

```
# /mnt/mlnx_add_kernel_support.sh --make-tgz <optional --kmp> -k $(uname -r) -m /mnt/
Note: This program will create MLNX_OFED_LINUX TGZ for rhel7.6 under /tmp directory.
Do you want to continue?[y/N]:y
See log file /tmp/mlnx_iso.4120_logs/mlnx_ofed_iso.4120.log

Checking if all needed packages are installed...
Building MLNX_OFED_LINUX RPMS . Please wait...
Creating metadata-rpms for 3.10.0-957.21.3.el7.x86_64 ...
WARNING: If you are going to configure this package as a repository, then please note
WARNING: that it contains unsigned rpms, therefore, you need to disable the gpgcheck
WARNING: by setting 'gpgcheck=0' in the repository conf file.
Created /tmp/MLNX_OFED_LINUX-5.2-0.5.5.0-rhel7.6-x86_64-ext.tgz
```

4. Open the tarball.

```
# cd /tmp/
# tar -xvf /tmp/MLNX_OFED_LINUX-5.2-0.5.5.0-rhel7.6-x86_64-ext.tgz
```

5. Create an apt-get repository configuration file called "/etc/apt/sources.list.d/mlnx_ofed.list" with the following content:

```
deb [trusted=yes] file:/<path to extracted MLNX_OFED package>/DEBS ./
```

6. Update the apt-get cache.

```
# sudo apt-get update
```

3.2.3.11.2.2 Installing MLNX_OFED Using the apt-get Tool

After setting up the apt-get repository for MLNX_OFED package, perform the following:

1. View the available package groups by invoking:

```
# apt-cache search mlnx-ofed-
apt-cache search mlnx-ofed  ........
knem-dkms - DKMS support for mlnx-ofed kernel modules
mlnx-ofed-kernel-dkms - DKMS support for mlnx-ofed kernel modules
mlnx-ofed-kernel-utils - Userspace tools to restart and tune mlnx-ofed kernel modules
mlnx-ofed-vma-vpi - MLNX_OFED vma-vpi installer package  (with DKMS support)
mlnx-ofed-kernel-only - MLNX_OFED kernel-only installer package  (with DKMS support)
mlnx-ofed-bluefield - MLNX_OFED bluefield installer package  (with DKMS support)
mlnx-ofed-hpc-user-only - MLNX_OFED hpc-user-only installer package  (User Space packages only)
mlnx-ofed-dpdk-user-only - MLNX_OFED dpdk-user-only installer package  (User Space packages only)
mlnx-ofed-all-exact - MLNX_OFED all installer package  (with DKMS support) (exact)
mlnx-ofed-all - MLNX_OFED all installer package  (with DKMS support)
mlnx-ofed-vma-vpi-user-only - MLNX_OFED vma-vpi-user-only installer package  (User Space packages only)
mlnx-ofed-eth-only-user-only - MLNX_OFED eth-only-user-only installer package  (User Space packages only)
mlnx-ofed-vma-user-only - MLNX_OFED vma-user-only installer package  (User Space packages only)
mlnx-ofed-hpc - MLNX_OFED hpc installer package  (with DKMS support)
mlnx-ofed-bluefield-user-only - MLNX_OFED bluefield-user-only installer package  (User Space packages only)
mlnx-ofed-dpdk - MLNX_OFED dpdk installer package  (with DKMS support)
mlnx-ofed-vma-eth-user-only - MLNX_OFED vma-eth-user-only installer package  (User Space packages only)
mlnx-ofed-all-user-only - MLNX_OFED all-user-only installer package  (User Space packages only)
mlnx-ofed-vma-eth - MLNX_OFED vma-eth installer package  (with DKMS support)
mlnx-ofed-vma - MLNX_OFED vma installer package  (with DKMS support)
mlnx-ofed-dpdk-upstream-libs-user-only - MLNX_OFED dpdk-upstream-libs-user-only installer package  (User
Space packages only)
mlnx-ofed-basic-user-only - MLNX_OFED basic-user-only installer package  (User Space packages only)
mlnx-ofed-basic-exact - MLNX_OFED basic installer package  (with DKMS support) (exact)
mlnx-ofed-basic - MLNX_OFED basic installer package  (with DKMS support)
mlnx-ofed-dpdk-upstream-libs - MLNX_OFED dpdk-upstream-libs installer package  (with DKMS support)
```

where:

mlnx-ofed-all	MLNX_OFED all installer package
mlnx-ofed-basic	MLNX_OFED basic installer package
mlnx-ofed-vma	MLNX_OFED vma installer package
mlnx-ofed-hpc	MLNX_OFED HPC installer package
mlnx-ofed-vma-eth	MLNX_OFED vma-eth installer package
mlnx-ofed-vma-vpi	MLNX_OFED vma-vpi installer package
knem-dkms	MLNX_OFED DKMS support for mlnx-ofed kernel modules
kernel-dkms	MLNX_OFED kernel-dkms installer package
kernel-only	MLNX_OFED kernel-only installer package
bluefield	MLNX_OFED bluefield installer package
mlnx-ofed-all-exact	MLNX_OFED mlnx-ofed-all-exact installer package
dpdk	MLNX_OFED dpdk installer package

mlnx-ofed-basic-exact	MLNX_OFED mlnx-ofed-basic-exact installer package
dpdk-upstream-libs	MLNX_OFED dpdk-upstream-libs installer package

2. Install the desired group.

```
apt-get install '<group name>'
```

Example:

```
apt-get install mlnx-ofed-all
```

> Installing MLNX_OFED using the “apt-get” tool does not automatically update the firmware.
> To update the firmware to the version included in MLNX_OFED package, run:
> `# apt-get install mlnx-fw-updater`
> OR:
> Update the firmware to the latest version available on NVIDIA's website as described in “Updating Firmware After Installation” section.

3.2.4 Uninstall

3.2.4.1 Uninstalling NVIDIA OFED Using the YUM and apt-get Tools

Use the script `/usr/sbin/ofed_uninstall.sh` to uninstall the NVIDIA OFED package. The script is part of the ofed-scripts RPM.

3.2.5 Updating Firmware After Installation

The firmware can be updated using one of the following methods.

3.2.5.1 Updating the Device Online

To update the device online on the machine from the NVIDIA site, use the following command line:

```
mlxfwmanager --online -u -d <device>
```

Example:

```
# mlxfwmanager --online -u -d 0000:01:00.0
Querying Mellanox devices firmware ...

Device #1:
----------

  Device Type:      ConnectX6
  Part Number:      MCX653106A-HDA_Ax
  Description:      ConnectX-6 VPI adapter card; HDR IB (200Gb/s) and 200GbE; dual-port QSFP56; PCIe4.0 x16; tall
bracket; ROHS R6
  PSID:             MT_0000000225
  PCI Device Name:  0000:01:00.0
```

```
Base MAC:          98039b970cc2
Versions:          Current         Available
   FW              20.26.4012      20.27.1016
   PXE             3.6.0101        3.5.0903
   UEFI            14.21.0016      14.20.0025

Status:            Up to date
```

3.2.5.2 Updating Firmware and FPGA Image on Innova IPsec Cards

The firmware and FPGA update package (mlnx-fw-updater) are installed under "/opt/mellanox/mlnx-fw-updater" folder.
The latest FW and FPGA update package can be downloaded from nvidia.com/en-us/networking/ Products → Adapters → Smart Adapters → Innova IPsec → Download tab.

> The current update package available on nvidia.com/en-us/networking/ does not support the script below. An update package that supports this script will become available in a future release.

You can run the following update script using one of the modes below:

```
/opt/mellanox/mlnx-fw-updater/mlnx_fpga_updater.sh
```

- With -u flag to provide URL to the software package (tarball). Example:

```
./mlnx_fpga_updater.sh -u http://www.mellanox.com/downloads/fpga/ipsec/Innova_IPsec_<version>.tgz
```

- With -t flag to provide the path to the downloaded tarball. Example:

```
./mlnx_fpga_updater.sh -t <Innova_IPsec_bundle_file.tgz>
```

- With -p flag to provide the path to the downloaded and extracted tarball. Example:

```
./mlnx_fpga_updater.sh -p <Innova_IPsec_extracted_bundle_directory>
```

For more information on the script usage, you can run mlnx_fpga_updater.sh -h.

> It is recommended to perform firmware and FPGA upgrade on Innova IPsec cards using this script only.

3.2.5.3 Updating the Device Manually

When running the mlnxofedinstall script with the '--without-fw-update' option or using an OEM card that you now wish to (manually) update firmware on your adapter card(s), perform the steps below. The following steps are also appropriate to burn newer firmware that was downloaded from the website (i.e., nvidia.com/en-us/networking/ → Support → Support → Firmware Download).

1. Get the device's PSID.

```
mlxfwmanager_pci | grep PSID
PSID:            MT_1210110019
```

2. Download the firmware BIN file from the website or the OEM website.

3. Burn the firmware.

```
mlxfwmanager_pci -i <fw_file.bin>
```

4. Reboot the device once the firmware burning is completed.

3.2.5.4 Updating the Device Firmware Automatically Upon System Boot

Firmware can be automatically updated upon system boot.
The firmware update package (mlnx-fw-updater) is installed in the "/opt/mellanox/mlnx-fw-updater" folder, and the openibd service script can invoke the firmware update process if requested on boot.
If the firmware is updated, the following message will be printed to the system's standard logging file:

```
fw_updater: Firmware was updated. Please reboot your system for the changes to take effect.
```

Otherwise, the following message will be printed:

```
fw_updater: Didn't detect new devices with old firmware.
```

Please note that this feature is disabled by default. To enable the automatic firmware update upon system boot, set the following parameter to "yes" "RUN_FW_UPDATER_ONBOOT=yes" in the openibd service configuration file "/etc/infiniband/openib.conf".
You can opt to exclude a list of devices from the automatic firmware update procedure. To do so, edit the configurations file "/opt/mellanox/mlnx-fw-updater/mlnx-fw-updater.conf" and provide a comma separated list of PCI devices to exclude from the firmware update.
Example:

```
MLNX_EXCLUDE_DEVICES="00:05.0,00:07.0"
```

3.2.6 UEFI Secure Boot

All kernel modules included in MLNX_OFED for RHEL and SLES are signed with x.509 key to support loading the modules when Secure Boot is enabled.

3.2.6.1 Enrolling NVIDIA's x.509 Public Key On your Systems

In order to support loading MLNX_OFED drivers when an OS supporting Secure Boot boots on a UEFI-based system with Secure Boot enabled, the NVIDIA x.509 public key should be added to the UEFI Secure Boot key database and loaded onto the system key ring by the kernel.
Follow these steps below to add the NVIDIA's x.509 public key to your system:

Prior to adding the NVIDIA's x.509 public key to your system, please make sure:

- The 'mokutil' package is installed on your system
- The system is booted in UEFI mode

1. Download the x.509 public key.

```
# wget http://www.mellanox.com/downloads/ofed/mlnx_signing_key_pub.der
```

As of version 23.04, the builds for SLES15 sp4 and sp5 are being singed with a newer signing key. The corresponding public key can be downloaded from https://www.mellanox.com/downloads/ofed/nv_nbu_kernel_signing_key_pub.der.

2. Add the public key to the MOK list using the mokutil utility.
   You will be asked to enter and confirm a password for this MOK enrollment request.

```
# mokutil --import mlnx_signing_key_pub.der
```

3. Reboot the system.

The pending MOK key enrollment request will be noticed by shim.efi and it will launch MokManager.efi to allow you to complete the enrollment from the UEFI console. You will need to enter the password you previously associated with this request and confirm the enrollment. Once done, the public key is added to the MOK list, which is persistent. Once a key is in the MOK list, it will be automatically propagated to the system key ring and subsequent will be booted when the UEFI Secure Boot is enabled.

To see what keys have been added to the system key ring on the current boot, install the 'keyutils' package and run: #keyctl list %:.system_keyring

3.2.6.2 Removing Signature from Kernel Modules

The signature can be removed from a signed kernel module using the 'strip' utility which is provided by the 'binutils' package.

```
# strip -g my_module.ko
```

The strip utility will change the given file without saving a backup. The operation can be undone only by resigning the kernel module. Hence, we recommend backing up a copy prior to removing the signature.

To remove the signature from the MLNX_OFED kernel modules:

1. Remove the signature.

```
# rpm -qa | grep -E "kernel-ib|mlnx-ofa_kernel|iser|srp|knem|mlnx-rds|mlnx-nfsrdma|mlnx-nvme|mlnx-rdma-rxe"
 | xargs rpm -ql | grep "\.ko$" | xargs strip -g
```

   Once the signature is removed, a message as the below will no longer be presented upon module loading:

```
"Request for unknown module key 'Mellanox Technologies signing key:
61feb074fc7292f958419386ffdd9d5ca999e403' err -11"
```

However, please note that a message similar to the following will be presented:

```
"my_module: module verification failed: signature and/or required key missing - tainting kernel"
```

This message is presented once, only upon first module boot that either has no signature or whose key is not in the kernel key ring. Therefore, this message may go unnoticed. Once the system is rebooted after unloading and reloading a kernel module, the message will appear (this message cannot be eliminated).

2. Update the initramfs on RHEL systems with the stripped modules.

```
mkinitrd /boot/initramfs-$(uname -r).img $(uname -r) --force
```

3.2.7 Performance Tuning

Depending on the application of the user's system, it may be necessary to modify the default configuration of network adapters based on the ConnectX® adapters. In case tuning is required, please refer to the Performance Tuning for NVIDIA Adapters Community post.

3.3 Features Overview and Configuration

It is recommended to enable the "above 4G decoding" BIOS setting for features that require a large amount of PCIe resources (e.g., SR-IOV with numerous VFs, PCIe Emulated Switch, Large BAR Requests).

The chapter contains the following sections:

- Ethernet Network
- InfiniBand Network
- Storage Protocols
- Virtualization
- Resiliency
- Docker Containers
- HPC-X
- Fast Driver Unload
- OVS Offload Using ASAP² Direct

3.3.1 Ethernet Network

The chapter contains the following sections:

- Ethernet Interface
- Quality of Service (QoS)
- Quantized Congestion Notification (QCN)
- Ethtool
- Checksum Offload

- Ignore Frame Check Sequence (FCS) Errors
- RDMA over Converged Ethernet (RoCE)
- Flow Control
- Explicit Congestion Notification (ECN)
- RSS Support
- Time-Stamping
- Flow Steering
- Wake-on-LAN (WoL)
- Hardware Accelerated 802.1ad VLAN (Q-in-Q Tunneling)
- VLAN Stripping in Linux Verbs
- Offloaded Traffic Sniffer
- Dump Configuration
- Local Loopback Disable
- Kernel Transport Layer Security (kTLS) Offloads
- IPsec Crypto Offload
- IPsec Full Offload
- MACsec Full Offload

3.3.1.1 Ethernet Interface

3.3.1.1.1 Port Type Management/VPI Cards Configuration

Ports of ConnectX-4 adapter cards and above can be individually configured to work as InfiniBand or Ethernet ports. By default, both VPI ports are initialized as InfiniBand ports. If you wish to change the port type, use the mlxconfig script after the driver is loaded.
For further information on how to set the port type, please refer to the MFT User Manual (nvidia.com/en-us/networking/ → Products → Software → InfiniBand/VPI Software → MFT—Firmware Tools)

3.3.1.1.2 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 help 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 Queue Pair (QP) attribute given in the QP context. Multiple QPs may be associated with the same counter set. If multiple QPs share the same counter, the counter value will represent the cumulative total.

3.3.1.1.2.1 RoCE Counters

- RoCE counters are available only through sysfs located under:
  - # /sys/class/infiniband/<device>/ports/*/hw_counters/
  - # /sys/class/infiniband/<device>/hw_counters/
  - # /sys/class/infiniband/<device>/ports/*/counters/

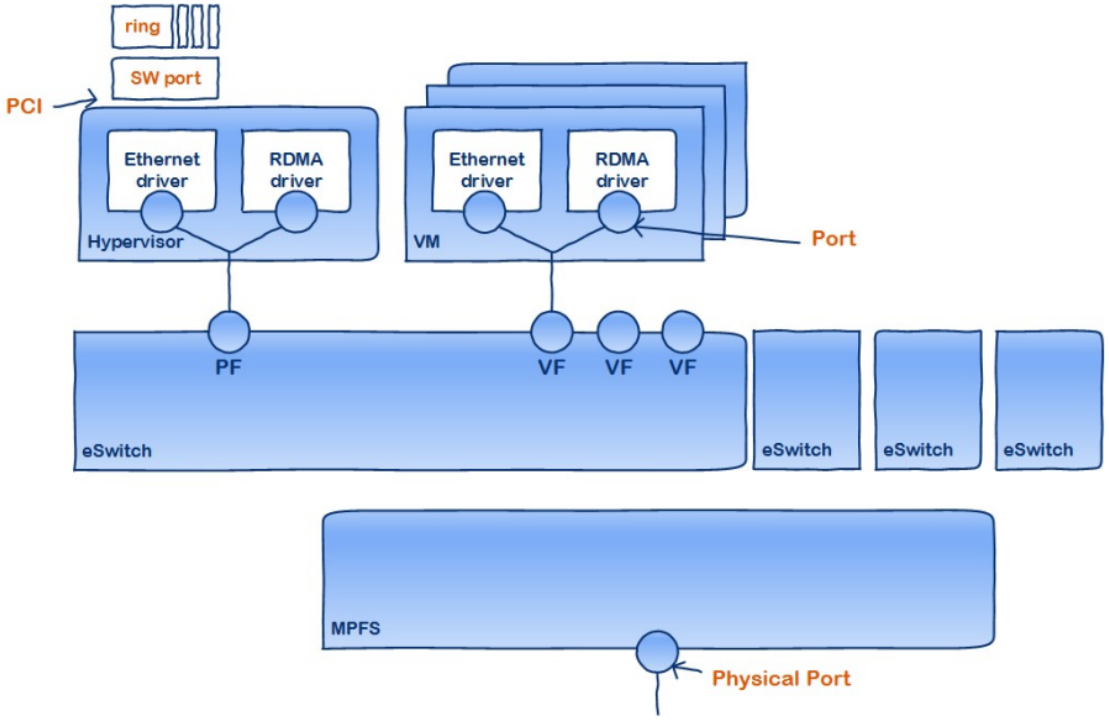
For mlx5 port and RoCE counters, refer to the Understanding mlx5 Linux Counters Community post.

3.3.1.1.2.2 SR-IOV Counters

Physical Function can also read Virtual Functions' port counters through sysfs located under # /sys/class/net/<interface_name>/device/sriov/<index>/stats/

3.3.1.1.2.3 ethtool Counters

The ethtool counters are counted in different places, according to which they are divided into groups. Each counters group may also have different counter types.



For the full list of supported ethtool counters, refer to the Understanding mlx5 ethtool Counters community post.

3.3.1.1.3 Persistent Naming

To avoid network interface renaming after boot or driver restart, create a file names "/etc/udev/rules.d/85-net-persistent-names.rules" with rules to set explicit interface names (the number, 85, must be more than 83).

- Example for Ethernet interfaces:

```
#PCI device 15b3:1019 (mlx5_core)
#NAME:="some name" , := is used to make sure that device name will be persistent.
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:02:c9:fa:c3:50", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="eth*", NAME:="eth1"
```

```
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:02:c9:fa:c3:51", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="eth*", NAME:="eth2"
```

- Example for IPoIB interfaces:

```
1. ATTR{type}=="32" is IPoIB interfaces
#2. Find address value for each IB interfaces:
#cat /sys/class/net/ib0/address | head -1 | cut -d":" -f"13-"
#24:8a:07:03:00:a5:29:38
#Use above output to place into ATTR{address}.
#Address match must start with ?* and only reference the last 8 bytes of the address
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{type}=="32", ATTR{address}=="?
*24:8a:07:03:00:a5:29:38", NAME:="ib0"
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{type}=="32", ATTR{address}=="?
*24:8a:07:03:00:a5:29:39", NAME:="ib1"
```

3.3.1.1.4 Interrupt Request (IRQ) Naming

Once IRQs are allocated by the driver, they are named `mlx5_comp<x>@pci:<pci_addr>`. The IRQ name is constant and is not affected by the interface state.
The mlx5_core driver allocates all IRQs during loading time to support the maximum possible number of channels. Once the driver is up, no further IRQs are freed or allocated. Changing the number of working channels does not re-allocate or free the IRQs.

3.3.1.2 Quality of Service (QoS)

Quality of Service (QoS) is a mechanism of assigning a priority to a network flow (socket, rdma_cm connection) and manage its guarantees, limitations and its priority over other flows. This is accomplished by mapping the user's priority to a hardware TC (traffic class) through a 2/3 stage process. The TC is assigned with the QoS attributes and the different flows behave accordingly.

3.3.1.2.1 Mapping Traffic to Traffic Classes

Mapping traffic to TCs consists of several actions which are user controllable, some controlled by the application itself and others by the system/network administrators.
The following is the general mapping traffic to Traffic Classes flow:

1. The application sets the required Type of Service (ToS).
2. The ToS is translated into a Socket Priority (sk_prio).
3. The sk_prio is mapped to a User Priority (UP) by the system administrator (some applications set sk_prio directly).
4. The UP is mapped to TC by the network/system administrator.
5. TCs hold the actual QoS parameters

QoS can be applied on the following types of traffic. However, the general QoS flow may vary among them:

- Plain Ethernet - Applications use regular inet sockets and the traffic passes via the kernel Ethernet driver
- RoCE - Applications use the RDMA API to transmit using Queue Pairs (QPs)
- Raw Ethernet QP - Application use VERBs API to transmit using a Raw Ethernet QP

3.3.1.2.2 Plain Ethernet Quality of Service Mapping

Applications use regular inet sockets and the traffic passes via the kernel Ethernet driver. The following is the Plain Ethernet QoS mapping flow:

1. The application sets the ToS of the socket using setsockopt (IP_TOS, value).
2. ToS is translated into the sk_prio using a fixed translation:

```
TOS 0 <=> sk_prio 0
TOS 8 <=> sk_prio 2
TOS 24 <=> sk_prio 4
TOS 16 <=> sk_prio 6
```

3. The Socket Priority is mapped to the UP in the following conditions:
    a. If the underlying device is a VLAN device, egress_map is used controlled by the vconfig command. This is per VLAN mapping.
    b. If the underlying device is not a VLAN device, the mapping is done in the driver.
4. The UP is mapped to the TC as configured by the mlnx_qos tool or by the lldpad daemon if DCBX is used.

⚠ Socket applications can use setsockopt (SK_PRIO, value) to directly set the sk_prio of the socket. In this case, the ToS to sk_prio fixed mapping is not needed. This allows the application and the administrator to utilize more than the 4 values possible via ToS.

⚠ In the case of a VLAN interface, the UP obtained according to the above mapping is also used in the VLAN tag of the traffic.

3.3.1.2.3 RoCE Quality of Service Mapping

Applications use RDMA-CM API to create and use QPs. The following is the RoCE QoS mapping flow:

1. The application sets the ToS of the QP using the rdma_set_option option(RDMA_OPTION_ID_TOS, value).
2. ToS is translated into the Socket Priority (sk_prio) using a fixed translation:

```
TOS 0 <=> sk_prio 0
TOS 8 <=> sk_prio 2
TOS 24 <=> sk_prio 4
TOS 16 <=> sk_prio 6
```

3. The Socket Priority is mapped to the User Priority (UP) using the tc command.
    - In the case of a VLAN device where the parent real device is used for the purpose of this mapping
    - If the underlying device is a VLAN device, and the parent real device was not used for the mapping, the VLAN device's egress_map is used
4. UP is mapped to the TC as configured by the mlnx_qos tool or by the lldpad daemon if DCBX is used.

⚠ With RoCE, there can only be 4 predefined ToS values for the purpose of QoS mapping.

3.3.1.2.4 Map Priorities with set_egress_map

For RoCE old kernels that do not support set_egress_map, use the tc_wrap script to map between sk_prio and UP. Use tc_wrap with option -u. For example:

```
tc_wrap -i <ethX> -u <skprio2up mapping>
```

3.3.1.2.5 Quality of Service Properties

The different QoS properties that can be assigned to a TC are:

- Strict Priority
- Enhanced Transmission Selection (ETS)
- Rate Limit
- Trust State
- Receive Buffer
- DCBX Control Mode

3.3.1.2.5.1 Strict Priority

When setting a TC's transmission algorithm to be 'strict', then this TC has absolute (strict) priority over other TC strict priorities coming before it (as determined by the TC number: TC 7 is the highest priority, TC 0 is lowest). It also has an absolute priority over nonstrict TCs (ETS).
This property needs to be used with care, as it may easily cause starvation of other TCs.
A higher strict priority TC is always given the first chance to transmit. Only if the highest strict priority TC has nothing more to transmit, will the next highest TC be considered.
Nonstrict priority TCs will be considered last to transmit.
This property is extremely useful for low latency low bandwidth traffic that needs to get immediate service when it exists, but is not of high volume to starve other transmitters in the system.

3.3.1.2.5.2 Enhanced Transmission Selection (ETS)

Enhanced Transmission Selection standard (ETS) exploits the time periods in which the offered load of a particular Traffic Class (TC) is less than its minimum allocated bandwidth by allowing the difference to be available to other traffic classes.
After servicing the strict priority TCs, the amount of bandwidth (BW) left on the wire may be split among other TCs according to a minimal guarantee policy.
If, for instance, TC0 is set to 80% guarantee and TC1 to 20% (the TCs sum must be 100), then the BW left after servicing all strict priority TCs will be split according to this ratio.
Since this is a minimum guarantee, there is no maximum enforcement. This means, in the same example, that if TC1 did not use its share of 20%, the reminder will be used by TC0.
ETS is configured using the mlnx_qos tool (mlnx_qos) which allows you to:

- Assign a transmission algorithm to each TC (strict or ETS)

- Set minimal BW guarantee to ETS TCs
  Usage:

```
mlnx_qos -i \[options\]
```

3.3.1.2.5.3 Rate Limit

Rate limit defines a maximum bandwidth allowed for a TC. Please note that 10% deviation from the requested values is considered acceptable.

3.3.1.2.5.4 Trust State

Trust state enables prioritizing sent/received packets based on packet fields.
The default trust state is PCP. Ethernet packets are prioritized based on the value of the field (PCP/DSCP).
For further information on how to configure Trust mode, please refer to HowTo Configure Trust State on NVIDIA Adapters community post.

> ⚠ Setting the Trust State mode shall be done before enabling SR-IOV in order to propagate the Trust State to the VFs.

3.3.1.2.5.5 Receive Buffer

By default, the receive buffer configuration is controlled automatically. Users can override the receive buffer size and receive buffer's xon and xoff thresholds using mlnx_qos tool.
For further information, please refer to HowTo Tune the Receive buffers on NVIDIA Adapters community post.

3.3.1.2.5.6 DCBX Control Mode

DCBX settings, such as "ETS" and "strict priority" can be controlled by firmware or software. When DCBX is controlled by firmware, changes of QoS settings cannot be done by the software. The DCBX control mode is configured using the mlnx_qos -d os/fw command.
For further information on how to configure the DCBX control mode, please refer to mlnx_qos community post.

3.3.1.2.6 Quality of Service Tools

3.3.1.2.6.1 mlnx_qos

mlnx_qos is a centralized tool used to configure QoS features of the local host. It communicates directly with the driver thus does not require setting up a DCBX daemon on the system.
The mlnx_qos tool enables the administrator of the system to:

- Inspect the current QoS mappings and configuration
  The tool will also display maps configured by TC and vconfig set_egress_map tools, in order to give a centralized view of all QoS mappings.
- Set UP to TC mapping
- Assign a transmission algorithm to each TC (strict or ETS)

- Set minimal BW guarantee to ETS TCs
- Set rate limit to TCs
- Set DCBX control mode
- Set cable length
- Set trust state

For an unlimited ratelimit, set the ratelimit to 0.

Usage

```
mlnx_qos -i <interface> \[options\]
```

Options

--version	Show the program's version number and exit
-h, --help	Show this help message and exit
-f LIST, --pfc=LIST	Set priority flow control for each priority. LIST is a comma separated value for each priority starting from 0 to 7. Example: 0,0,0,0,1,1,1,1 enable PFC on TC4-7
-p LIST, --prio_tc=LIST	Maps UPs to TCs. LIST is 8 comma-separated TC numbers. Example: 0,0,0,0,1,1,1,1 maps UPs 0-3 to TC0, and UPs 4-7 to TC1
-s LIST, --tsa=LIST	Transmission algorithm for each TC. LIST is comma separated algorithm names for each TC. Possible algorithms: strict, ets and vendor. Example: vendor,strict,ets,ets,ets,ets,ets,ets sets TC0 to vendor, TC1 to strict, TC2-7 to ets
-t LIST, --tcbw=LIST	Set the minimally guaranteed %BW for ETS TCs. LIST is comma-separated percents for each TC. Values set to TCs that are not configured to ETS algorithm are ignored but must be present. Example: if TC0,TC2 are set to ETS, then 10,0,90,0,0,0,0,0will set TC0 to 10% and TC2 to 90%. Percents must sum to 100
-r LIST, --ratelimit=LIST	Rate limit for TCs (in Gbps). LIST is a comma-separated Gbps limit for each TC. Example: 1,8,8 will limit TC0 to 1Gbps, and TC1,TC2 to 8 Gbps each
-d DCBX, --dcbx=DCBX	Set dcbx mode to firmware controlled(fw) or OS controlled(os). Note, when in OS mode, mlnx_qos should not be used in parallel with other dcbx tools, such as lldptool
--trust=TRUST	set priority trust state to pcp or dscp
--dscp2prio=DSCP2PRIO	Set/del a (dscp,prio) mapping. Example 'set,30,2' maps dscp 30 to priority 2. 'del,30,2' resets the dscp 30 mapping back to the default setting priority 0

--cable_len=CABLE_LEN	Set cable_len for buffer's xoff and xon thresholds
-i INTF, --interface=INTF	Interface name
-a	Show all interface's TCs

Get Current Configuration

```
ofed_scripts/utils/mlnx_qos -i ens1f0
DCBX mode: OS controlled
Priority trust state: dscp
dscp2prio mapping:
        prio:0 dscp:07,06,05,04,03,02,01,00,
        prio:1 dscp:15,14,13,12,11,10,09,08,
        prio:2 dscp:23,22,21,20,19,18,17,16,
        prio:3 dscp:31,30,29,28,27,26,25,24,
        prio:4 dscp:39,38,37,36,35,34,33,32,
        prio:5 dscp:47,46,45,44,43,42,41,40,
        prio:6 dscp:55,54,53,52,51,50,49,48,
        prio:7 dscp:63,62,61,60,59,58,57,56,
Cable len: 7
PFC configuration:
        priority 0 1 2 3 4 5 6 7
        enabled 0 0 0 0 0 0 0 0
tc: 0 ratelimit: unlimited, tsa: vendor
         priority: 1
tc: 1 ratelimit: unlimited, tsa: vendor
         priority: 0
tc: 2 ratelimit: unlimited, tsa: vendor
         priority: 2
tc: 3 ratelimit: unlimited, tsa: vendor
         priority: 3
tc: 4 ratelimit: unlimited, tsa: vendor
         priority: 4
tc: 5 ratelimit: unlimited, tsa: vendor
         priority: 5
tc: 6 ratelimit: unlimited, tsa: vendor
         priority: 6
tc: 7 ratelimit: unlimited, tsa: vendor
         priority: 7
```

Set ratelimit. 3Gbps for tc0 4Gbps for tc1 and 2Gbps for tc2

```
# mlnx_qos -i <interface> -p 0,1,2 -r 3,4,2
tc: 0 ratelimit: 3 Gbps, tsa: strict
         up:  0
                 skprio: 0
                 skprio: 1
                 skprio: 2 (tos: 8)
                 skprio: 3
                 skprio: 4 (tos: 24)
                 skprio: 5
                 skprio: 6 (tos: 16)
                 skprio: 7
                 skprio: 8
                 skprio: 9
                 skprio: 10
                 skprio: 11
                 skprio: 12
                 skprio: 13
                 skprio: 14
                 skprio: 15
         up:  3
         up:  4
         up:  5
         up:  6
         up:  7
tc: 1 ratelimit: 4 Gbps, tsa: strict
         up:  1
tc: 2 ratelimit: 2 Gbps, tsa: strict
         up:  2
```

ConfigureQoS. Map UP0,7 to tc0,1,2,3 to tc1 and 4,5,6 to tc
2. Set tc0,tc1 as ets and tc2 as strict. Divide ets 30% for tc0 and 70% for tc1

```
# mlnx_qos -i <interface> -s ets,ets,strict -p 0,1,1,1,2,2,2 -t 30,70
tc: 0 ratelimit: 3 Gbps, tsa: ets, bw: 30%
         up:  0
                 skprio: 0
                 skprio: 1
                 skprio: 2 (tos: 8)
                 skprio: 3
                 skprio: 4 (tos: 24)
                 skprio: 5
                 skprio: 6 (tos: 16)
                 skprio: 7
                 skprio: 8
                 skprio: 9
                 skprio: 10
                 skprio: 11
                 skprio: 12
                 skprio: 13
                 skprio: 14
                 skprio: 15
  up:  7
tc: 1 ratelimit: 4 Gbps, tsa: ets, bw: 70%
         up:  1
         up:  2
         up:  3
tc: 2 ratelimit: 2 Gbps, tsa: strict
         up:  4
         up:  5
         up:  6
```

tc and tc_wrap.py

The tc tool is used to create 8 Traffic Classes (TCs).
The tool will either use the sysfs (/sys/class/net/<ethX>/qos/tc_num) or the tc tool to create the TCs.

Usage

```
tc_wrap.py -i <interface> \[options\]
```

Options

--version	show program's version number and exit
-h, --help	show this help message and exit
-u SKPRIO_UP, --skprio_up=SKPRIO_UP	maps sk_prio to priority for RoCE. LIST is <=16 comma separated priority. index of element is sk_prio
-i INTF, --interface=INTF	Interface name

Example
Run:

```
tc_wrap.py -i enp139s0
```

Output:

```
Tarrfic classes are set to 8

UP  0
    skprio: 0 (vlan 5)
UP  1
    skprio: 1 (vlan 5)
UP  2
    skprio: 2 (vlan 5 tos: 8)
UP  3
    skprio: 3 (vlan 5)
```

```
UP  4
    skprio: 4 (vlan 5 tos: 24)
UP  5
    skprio: 5 (vlan 5)
UP  6
    skprio: 6 (vlan 5 tos: 16)
UP  7
    skprio: 7 (vlan 5)
```

3.3.1.2.6.2 Additional Tools

tc tool compiled with the sch_mqprio module is required to support kernel v2.6.32 or higher. This is a part of iproute2 package v2.6.32-19 or higher. Otherwise, an alternative custom sysfs interface is available.

- mlnx_qos tool (package: ofed-scripts) requires python version 2.5 < = X
- tc_wrap.py (package: ofed-scripts) requires python version 2.5 < = X

3.3.1.3 Quantized Congestion Notification (QCN)

Congestion control is used to reduce packet drops in lossy environments and mitigate congestion spreading and resulting victim flows in lossless environments.
The Quantized Congestion Notification (QCN) IEEE standard (802.1Qau) provides congestion control for long-lived flows in limited bandwidth-delay product Ethernet networks. It is part of the IEEE Data Center Bridging (DCB) protocol suite, which also includes ETS, PFC, and DCBX. QCN in conducted at L2, and is targeted for hardware implementations. QCN applies to all Ethernet packets and all transports, and both the host and switch behavior is detailed in the standard.
QCN user interface allows the user to configure QCN activity. QCN configuration and retrieval of information is done by the mlnx_qcn tool. The command interface provides the user with a set of changeable attributes, and with information regarding QCN's counters and statistics. All parameters and statistics are defined per port and priority. QCN command interface is available if and only the hardware supports it.

3.3.1.3.1 QCN Tool - mlnx_qcn

> ⚠ This tool is supported on ConnectX-3 and ConnectX-3 Pro NICs only.

mlnx_qcn is a tool used to configure QCN attributes of the local host. It communicates directly with the driver thus does not require setting up a DCBX daemon on the system.

The mlnx_qcn enables the user to:

- Inspect the current QCN configurations for a certain port sorted by priority
- Inspect the current QCN statistics and counters for a certain port sorted by priority
- Set values of chosen QCN parameters

Usage

```
mlnx_qcn -i <interface> \[options\]
```

Options

--version	Show program's version number and exit

-h, --help	Show this help message and exit
-i INTF, --interface=INTF	Interface name
-g TYPE, --get_type=TYPE	Type of information to get statistics/parameters
--rpg_enable=RPG_ENABLE_LIST	Set value of rpg_enable according to priority, use spaces between values and -1 for unknown values.
--rppp_max_rps=RPPP_MAX_RPS_LIST	Set value of rppp_max_rps according to priority, use spaces between values and -1 for unknown values.
--rpg_time_reset=RPG_TIME_RESET_LIST	Set value of rpg_time_reset according to priority, use spaces between values and -1 for unknown values.
--rpg_byte_reset=RPG_BYTE_RESET_LIST	Set value of rpg_byte_reset according to priority, use spaces between values and -1 for unknown values.
--rpg_threshold=RPG_THRESHOLD_LIST	Set value of rpg_threshold according to priority, use spaces between values and -1 for unknown values.
--rpg_max_rate=RPG_MAX_RATE_LIST	Set value of rpg_max_rate according to priority, use spaces between values and -1 for unknown values.
--rpg_ai_rate=RPG_AI_RATE_LIST	Set value of rpg_ai_rate according to priority, use spaces between values and -1 for unknown values.
--rpg_hai_rate=RPG_HAI_RATE_LIST	Set value of rpg_hai_rate according to priority, use spaces between values and -1 for unknown values.
--rpg_gd=RPG_GD_LIST	Set value of rpg_gd according to priority, use spaces between values and -1 for unknown values.
--rpg_min_dec_fac=RPG_MIN_DEC_FAC_LIST	Set value of rpg_min_dec_fac according to priority, use spaces between values and -1 for unknown values.
--rpg_min_rate=RPG_MIN_RATE_LIST	Set value of rpg_min_rate according to priority, use spaces between values and -1 for unknown values.
--cndd_state_machine=CNDD_STATE_MACHINE_LIST	Set value of cndd_state_machine according to priority, use spaces between values and -1 for unknown values.

To get QCN current configuration sorted by priority, run:

```
mlnx_qcn -i eth2 -g parameters
```

To show QCN's statistics sorted by priority, run:

```
mlnx_qcn -i eth2 -g statistics
```

Output example of running mlnx_qcn -i eth2 -g parameters:

```
priority 0:
rpg_enable: 0
rppp_max_rps: 1000
rpg_time_reset: 1464
rpg_byte_reset: 150000
rpg_threshold: 5
rpg_max_rate: 40000
rpg_ai_rate: 10
rpg_hai_rate: 50
rpg_gd: 8
rpg_min_dec_fac: 2
rpg_min_rate: 10
cndd_state_machine: 0
priority 1:
rpg_enable: 0
rppp_max_rps: 1000
rpg_time_reset: 1464
rpg_byte_reset: 150000
rpg_threshold: 5
rpg_max_rate: 40000
rpg_ai_rate: 10
rpg_hai_rate: 50
rpg_gd: 8
rpg_min_dec_fac: 2
rpg_min_rate: 10
cndd_state_machine: 0
.............................
.............................
priority 7:
rpg_enable: 0
rppp_max_rps: 1000
rpg_time_reset: 1464
rpg_byte_reset: 150000
rpg_threshold: 5
rpg_max_rate: 40000
rpg_ai_rate: 10
rpg_hai_rate: 50
rpg_gd: 8
rpg_min_dec_fac: 2
rpg_min_rate: 10
cndd_state_machine: 0
```

3.3.1.3.2 Setting QCN Configuration

Setting QCN parameters requires updating its value for each priority. '-1' indicates no change in the current value.

Example for setting 'rpg_enable' in order to enable QCN for priorities 3, 5, 6:

```
mlnx_qcn -i eth2 --rpg_enable=-1 -1 -1 1 -1 1 1 -1
```

Example for setting 'rpg_hai_rate' for priorities 1, 6, 7:

```
mlnx_qcn -i eth2 --rpg_hai_rate=60 -1 -1 -1 -1 -1 60 60
```

3.3.1.4 Ethtool

Ethtool is a standard Linux 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, auto-negotiation and flow control for Ethernet devices
- Control checksum offload and other hardware offload features
- Control DMA ring sizes and interrupt moderation

- Flash device firmware using a .mfa2 image

Ethtool Supported Options

Options	Description
ethtool --set-priv-flags eth<x> <priv flag> <on/off>	Enables/disables driver feature matching the given private flag.
ethtool --show-priv-flags eth<x>	Shows driver private flags and their states (ON/OFF).
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 -c eth<x>	Queries interrupt coalescing settings.
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.
ethtool -C eth<x> [rx-usecs N] [rx-frames N]	Sets the interrupt coalescing setting. rx-frames will be enforced immediately, rx-usecs will be enforced only when 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 -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> adaptive-tx on|off	Note: Supported by mlx5e for ConnectX-4 and above adapter cards. Enables/disables adaptive interrupt moderation. By default, the driver uses adaptive interrupt moderation for the transmit path, which adjusts the moderation parameters (time/frames) to the traffic pattern.
ethtool -g eth<x>	Queries the ring size values.
ethtool -G eth<x> [rx <N>] [tx <N>]	Modifies the ring size.
ethtool -i eth<x>	Checks driver and device information. For example: driver: mlx5_core version: 5.1-0.4.0 firmware-version: 4.6.4046 (MT_QEMU000000) expansion-rom-version: bus-info: 0000:07:00.0 supports-statistics: yes supports-test: yes supports-eeprom-access: no supports-register-dump: no supports-priv-flags: yes
ethtool -k eth<x>	Queries the stateless offload status.

Options	Description
ethtool -K eth<x> [rx on|off] [tx on|off] [sg on|off] [tso on|off] [lro on|off] [gro on|off] [gso on|off] [rxvlan on|off] [txvlan on|off] [ntuple on/off] [rxhash on/off] [rx-all on/off] [rx-fcs on/off]	Sets the stateless offload status. TCP Segmentation Offload (TSO), Generic Segmentation Offload (GSO): increase outbound throughput by reducing CPU overhead. It works by queuing up large buffers and letting the network interface card split them into separate packets. Large Receive Offload (LRO): increases inbound throughput of high-bandwidth network connections by reducing CPU overhead. It works by aggregating multiple incoming packets from a single stream into a larger buffer before they are passed higher up the networking stack, thus reducing the number of packets that have to be processed. LRO is available in kernel versions < 3.1 for untagged traffic. Hardware VLAN insertion Offload (txvlan): When enabled, the sent VLAN tag will be inserted into the packet by the hardware. **Note**: LRO will be done whenever possible. Otherwise GRO will be done. Generic Receive Offload (GRO) is available throughout all kernels. Hardware VLAN Striping Offload (rxvlan): When enabled received VLAN traffic will be stripped from the VLAN tag by the hardware. RX FCS (rx-fcs): Keeps FCS field in the received packets.Sets the stateless offload status. RX FCS validation (rx-all): Ignores FCS validation on the received packets.
ethtool -l eth<x>	Shows the number of channels.
ethtool -L eth<x> [rx <N>] [tx <N>]	Sets the number of channels. **Notes**: • This also resets the RSS table to its default distribution, which is uniform across the cores on the NUMA (non-uniform memory access) node that is closer to the NIC. • For ConnectX®-4 cards, use ethtool -L eth<x> combined <N> to set both RX and TX channels.
ethtool -m|--dump-module-eeprom eth<x> [raw on|off] [hex on|off] [offset N] [length N]	Queries/decodes the cable module eeprom information.
ethtool -p|--identify DEVNAME	Enables visual identification of the port by LED blinking [TIME-IN-SECONDS].
ethtool -p|--identify eth<x> <LED duration>	Allows users to identify interface's physical port by turning the ports LED on for a number of seconds. **Note**: The limit for the LED duration is 65535 seconds.
ethtool -S eth<x>	Obtains additional device statistics.

<table>
<tr><th>Options</th><th>Description</th></tr>
<tr><td>ethtool -s eth<x> advertise <N> autoneg on</td><td>Changes the advertised link modes to requested link modes <N>
To check the link modes' hex values, run <code><man ethtool></code> and to check the supported link modes, run <code>ethtool eth<x></code>
For advertising new link modes, make sure to configure the entire bitmap as follows:
<table><tr><td>200GAUI-4 / 200GBASE-CR4/KR4</td><td>0x7c000000000000000</td></tr><tr><td>100GAUI-2 / 100GBASE-CR2 / KR2</td><td>0x3E00000000000000</td></tr><tr><td>CAUI-4 / 100GBASE-CR4 / KR4</td><td>0xF000000000</td></tr><tr><td>50GAUI-1 / LAUI-1/ 50GBASE-CR / KR</td><td>0x1F0000000000000</td></tr><tr><td>50GAUI-2 / LAUI-2/ 50GBASE-CR2/KR2</td><td>0x10C00000000</td></tr><tr><td>XLAUI-4/XLPPI-4 // 40G</td><td>0x7800000</td></tr><tr><td>25GAUI-1/ 25GBASE-CR / KR</td><td>0x380000000</td></tr><tr><td>XFI / XAUI-1 // 10G</td><td>0x7C0000181000</td></tr><tr><td>5GBASE-R</td><td>0x1000000000000</td></tr><tr><td>2.5GBASE-X / 2.5GMII</td><td>0x820000000000</td></tr><tr><td>1000BASE-X / SGMII</td><td>0x20000020020</td></tr></table>
Notes:
Both previous and new link modes configurations are supported, however, they must be run separately.
Any link mode configuration on Kernels below v5.1 and ConnectX-6 HCAs will result in the advertisement of the full capabilities.
<code><autoneg on></code> only sends a hint to the driver that the user wants to modify advertised link modes and not speed.</td></tr>
<tr><td>ethtool -s eth<x> msglvl [N]</td><td>Changes the current driver message level.</td></tr>
<tr><td>ethtool -s eth<x> speed <SPEED> autoneg off</td><td>Changes the link speed to requested <SPEED>. To check the supported speeds, run <code>ethtool eth<x></code>.
Note: <autoneg off> does not set autoneg OFF, it only hints the driver to set a specific speed.</td></tr>
<tr><td>ethtool -t eth<x></td><td>Performs a self-diagnostics test.</td></tr>
<tr><td>ethtool -T eth<x></td><td>Shows time stamping capabilities</td></tr>
<tr><td>ethtool -x eth<x></td><td>Retrieves the receive flow hash indirection table.</td></tr>
<tr><td>ethtool -X eth<x> equal a b c...</td><td>Sets the receive flow hash indirection table.
Note: The RSS table configuration is reset whenever the number of channels is modified (using ethtool -L command).</td></tr>
<tr><td>ethtool --show-fec eth<x></td><td>Queries current Forward Error Correction (FEC) encoding in case FEC is supported.
Note: An output of "baser" implies Firecode encoding.</td></tr>
<tr><td>ethtool --set-fec eth<x> encoding auto|off|rs|baser</td><td>Configures Forward Error Correction (FEC).
Note: 'baser' encoding applies to the Firecode encoding, and 'auto' regards the HCA's default.</td></tr>
<tr><td>ethtool -f|--flash <devname> FILE [N]</td><td>Flash firmware image on the device using the specified .mfa2 file (FILE). By default, the command flashes all the regions on the device unless a region number (N) is specified.</td></tr>
</table>

3.3.1.5 Checksum Offload

The following Receive IP/L4 Checksum Offload modes are supported.

- CHECKSUM_UNNECESSARY: When this mode is used, 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_COMPLETE: When this mode is used, 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_NONE: When this mode is used, the driver indicates to the Linux Networking Stack that it must calculate and validate the IP/L4 checksum.

3.3.1.6 Ignore Frame Check Sequence (FCS) Errors

Upon receiving packets, the packets go through a checksum validation process for the FCS field. If the validation fails, the received packets are dropped.
When FCS is enabled (disabled by default), the device does not validate the FCS field even if the field is invalid.
It is not recommended to enable FCS.
For further information on how to enable/disable FCS, please refer to ethtool option rx-fcs on/off.

3.3.1.7 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® Ethernet adapter cards family with RoCE uses the proven and efficient RDMA transport to provide the platform for deploying RDMA technology in mainstream data center application at 10GigE and 40GigE link-speed. ConnectX® Ethernet adapter cards family with its hardware offload support takes advantage of this efficient RDMA transport (InfiniBand) services over Ethernet to deliver ultra-low latency for performance-critical and transaction-intensive applications such as financial, database, storage, and content delivery networks.
When working with RDMA applications over Ethernet link layer the following points should be noted:

- The presence of a Subnet Manager (SM) is not required in the fabric. Thus, operations that require communication with the SM are managed in a different way in RoCE. This does not affect the API but only the actions such as joining the multicast group, that need to be taken when using the API
- Since LID is a layer 2 attribute of the InfiniBand protocol stack, it is not set for a port and is displayed as zero when querying the port
- With RoCE, the alternate path is not set for RC QP. Therefore, APM (another type of High Availability and part of the InfiniBand protocol) is not supported

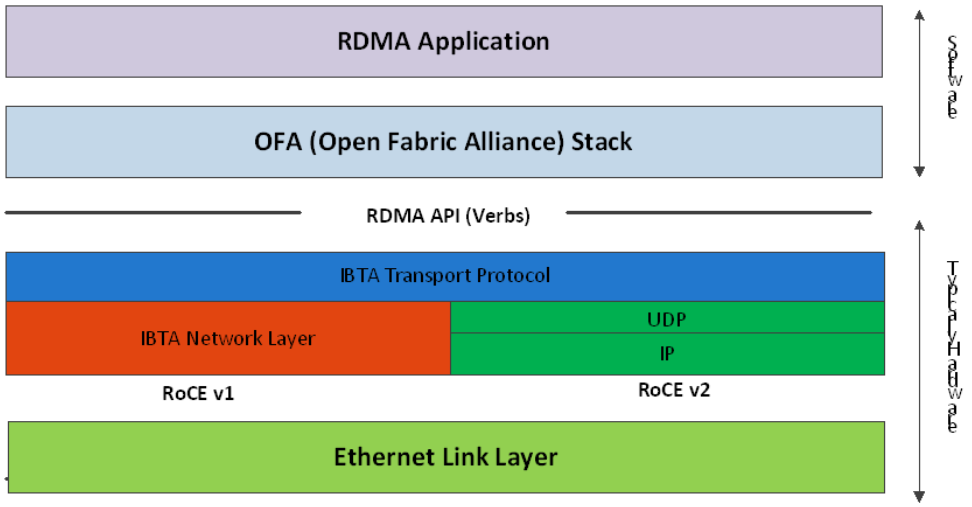
- Since the SM is not present, querying a path is impossible. Therefore, the path record structure must be filled with relevant values before establishing a connection. Hence, it is recommended working with RDMA-CM to establish a connection as it takes care of filling the path record structure
- VLAN tagged Ethernet frames carry a 3-bit priority field. The value of this field is derived from the IB SL field by taking the 3 least significant bits of the SL field
- RoCE traffic is not shown in the associated Ethernet device's counters since it is offloaded by the hardware and does not go through Ethernet network driver. RoCE traffic is counted in the same place where InfiniBand traffic is counted; /sys/class/infiniband/<device>/ports/<port number>/counters/

3.3.1.7.1 RoCE Modes

RoCE encapsulates IB transport in one of the following Ethernet packets:

- RoCEv1 - dedicated ether type (0x8915)
- RoCEv2 - UDP and dedicated UDP port (4791)

RoCEv1 and RoCEv2 Protocol Stack



3.3.1.7.1.1 RoCEv1

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

3.3.1.7.1.2 RoCEv2

A straightforward extension of the RoCE protocol enables traffic to operate in IP layer 3 environments. This capability is obtained via a simple modification of the RoCE packet format. Instead of the GRH used in RoCE, IP routable RoCE packets carry an IP header which allows traversal

of IP L3 Routers and a UDP header (RoCEv2 only) that serves as a stateless encapsulation layer for the RDMA Transport Protocol Packets over IP.

The proposed RoCEv2 packets use a well-known UDP destination port value that unequivocally distinguishes the datagram. Similar to other protocols that use UDP encapsulation, the UDP source port field is used to carry an opaque flow-identifier that allows network devices to implement packet forwarding optimizations (e.g. ECMP) while staying agnostic to the specifics of the protocol header format.
Furthermore, since this change exclusively affects the packet format on the wire, and due to the fact that with RDMA semantics packets are generated and consumed below the AP, applications can seamlessly operate over any form of RDMA service, in a completely transparent way.

Both RoCEv1 and RoCEv2 are supported by default; the driver associates all GID indexes to RoCEv1 and RoCEv2, thus, a single entry for each RoCE version.

For further information, please refer to Recommended Network Configuration Examples For RoCE Deployment Community post.

3.3.1.7.2 GID Table Population

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

- GID value
- GID type
- Network device

The GID table is occupied with two GIDs, both with the same GID value but with different types. The network device in an entry is the Ethernet device with the IP address that GID is associated with. The GID format can be of 2 types; IPv4 and IPv6. IPv4 GID is an IPv4-mapped IPv6 address, while IPv6 GID is the IPv6 address itself. Layer 3 header for packets associated with IPv4 GIDs will be IPv4 (for RoCEv2) and IPv6/GRH for packets associated with IPv6 GIDs and IPv4 GIDs for RoCEv1.

GID Table in sysfs

GID table is exposed to userspace via sysfs

- GID values can be read from:

```
/sys/class/infiniband/{device}/ports/{port}/gids/{index}
```

- GID type can be read from:

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

- GID net_device can be read from:

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

3.3.1.7.2.1 Setting the RoCE Mode for a Queue Pair (QP)

Setting RoCE mode for devices that support two RoCE modes is different for RC/UC QPs (connected QP types) and UD QP.

To modify an RC/UC QP (connected QP) from INIT to RTR, an Address Vector (AV) must be given. The AV, among other attributes, should specify the index of the port's GID table for the source GID of the QP. The GID type in that index will be used to set the RoCE type of the QP.

3.3.1.7.2.2 Setting RoCE Mode of RDMA_CM Applications

RDMA_CM interface requires only the active side of the peer to pass the IP address of the passive side. The RDMA_CM decides upon the source GID to be used and obtains it from the GID table. Since more than one instance of the GID value is possible, the lookup should be also according to the GID type. The type to use for the lookup is defined as a global value of the RDMA_CM module. Changing the value of the GID type for the GID table lookups is done using the cma_roce_mode script.

- *To print the current RoCE mode for a device port:*

```
cma_roce_mode -d <dev> -p <port>
```

- *To set the RoCE mode for a device port:*

```
cma_roce_mode -d <dev> -p <port> -m <1|2>
```

3.3.1.7.2.3 GID Table Example

The following is an example of the GID table.

DEV	PORT	INDEX	GID	IPv4	Type	Netdev
mlx5_0	1	0	fe80:0000:0000:0000:ba59:9fff:fe1a:e3ea		v1	p4p1
mlx5_0	1	1	fe80:0000:0000:0000:ba59:9fff:fe1a:e3ea		v2	p4p1
mlx5_0	1	2	0000:0000:0000:0000:0000:ffff:0a0a:0a01	10.10.10.1	v1	p4p1
mlx5_0	1	3	0000:0000:0000:0000:0000:ffff:0a0a:0a01	10.10.10.1	v2	p4p1
mlx5_1	1	0	fe80:0000:0000:0000:ba59:9fff:fe1a:e3eb		v1	p4p2
mlx5_1	1	1	fe80:0000:0000:0000:ba59:9fff:fe1a:e3eb		v2	p4p2

where:

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

3.3.1.7.3 RoCE Lossless Ethernet Configuration

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.

3.3.1.7.3.1 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 SwitchX User Manual

3.3.1.7.4 Installing and Loading the Driver

To install and load the driver:

1. Install MLNX_OFED (See Installation section for further details).
   RoCE is installed as part of mlx5 and other modules upon driver's installation.

   The list of the modules that will be loaded automatically upon boot can be found in the configuration file /etc/infiniband/openib.conf.

2. Query for the device's information. Example:

```
ibv_devinfo MLNX_OFED_LINUX-5.0-2.1.8.0:
```

3. Display the existing MLNX_OFED version.

```
ofed_info -s
hca_id: mlx5_0
        transport:                      InfiniBand (0)
        fw_ver:                         16.28.0578
        node_guid:                      ec0d:9a03:0044:3764
        sys_image_guid:                 ec0d:9a03:0044:3764
        vendor_id:                      0x02c9
        vendor_part_id:                 4121
        hw_ver:                         0x0
        board_id:                       MT_0000000009
        phys_port_cnt:                  1
                port:   1
                        state:                  PORT_ACTIVE (4)
                        max_mtu:                4096 (5)
                        active_mtu:             1024 (3)
                        sm_lid:                 0
                        port_lid:               0
                        port_lmc:               0x00
                        link_layer:             Ethernet
```

Output Notes:

The port's state is: Ethernet is in PORT_ACTIVE state	The port state can also be obtained by running the following command: # cat /sys/class/infiniband/mlx5_0/ports/1/state 4: ACTIVE

link_layer parameter shows that port 1 is Ethernet	The link_layer can also be obtained by running the following command: `# cat /sys/class/infiniband/mlx5_0/ports/1/link_layer Ethernet`
The fw_ver parameter shows that the firmware version is 16.28.0578.	The firmware version can also be obtained by running the following command: `# cat /sys/class/infiniband/mlx5_0/fw_ver 16.28.0578`

3.3.1.7.4.1 Associating InfiniBand Ports to Ethernet Ports

The mlx5_ib driver holds a reference to the net device for getting notifications about the state of the port, as well as using the mlx5_core driver to resolve IP addresses to MAC that are required for address vector creation. However, RoCE traffic does not go through the mlx5_core driver; it is completely offloaded by the hardware.

```
# ibdev2netdev
mlx5_0 port 1 <===> eth2
#
```

3.3.1.7.4.2 Configuring an IP Address to the netdev Interface

To configure an IP address to the netdev interface:

1. Configure an IP address to the netdev interface on both sides of the link.

```
# ifconfig eth2 20.4.3.220
# ifconfig eth2
eth2      Link encap:Ethernet  HWaddr 00:02:C9:08:E8:11
          inet addr:20.4.3.220  Bcast:20.255.255.255  Mask:255.0.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

2. Make sure that ping is working.

```
ping 20.4.3.219
PING 20.4.3.219 (20.4.3.219) 56(84) bytes of data.
64 bytes from 20.4.3.219: icmp_seq=1 ttl=64 time=0.873 ms
64 bytes from 20.4.3.219: icmp_seq=2 ttl=64 time=0.198 ms
64 bytes from 20.4.3.219: icmp_seq=3 ttl=64 time=0.167 ms
20.4.3.219 ping statistics —
3 packets transmitted, 3 received, 0% packet loss, time 2000ms rtt min/avg/max/mdev = 0.167/0.412/0.873/
0.326 ms
```

3.3.1.7.4.3 Adding VLANs

To add VLANs:

1. Make sure that the 8021.q module is loaded.

```
modprobe 8021q
```

2. Add VLAN.

```
# vconfig add eth2 7
Added VLAN with VID == 7 to IF -:eth2:-
#
```

3. Configure an IP address.

```
ifconfig eth2.7 7.4.3.220
```

3.3.1.7.4.4 Defining Ethernet Priority (PCP in 802.1q Headers)

1. Define Ethernet priority on the server.

```
# ibv_rc_pingpong -g 1 -i 2 -l 4
local address: LID 0x0000, QPN 0x1c004f, PSN 0x9daf6c, GID fe80::202:c900:708:e799
remote address: LID 0x0000, QPN 0x1c004f, PSN 0xb0a49b, GID fe80::202:c900:708:e811
8192000 bytes in 0.01 seconds = 4840.89 Mbit/sec
1000 iters in 0.01 seconds = 13.54 usec/iter
```

2. Define Ethernet priority on the client.

```
# ibv_rc_pingpong -g 1 -i 2 -l 4 sw419
local address: LID 0x0000, QPN 0x1c004f, PSN 0xb0a49b, GID fe80::202:c900:708:e811
remote address: LID 0x0000, QPN 0x1c004f, PSN 0x9daf6c, GID fe80::202:c900:708:e799
8192000 bytes in 0.01 seconds = 4855.96 Mbit/sec
1000 iters in 0.01 seconds = 13.50 usec/iter
```

3.3.1.7.4.5 Using rdma_cm Tests

1. Use rdma_cm test on the server.

```
# ucmatose
cmatose: starting server
initiating data transfers
completing sends
receiving data transfers
data transfers complete
cmatose: disconnecting
disconnected
test complete
return status 0
#
```

2. Use rdma_cm test on the client.

```
# ucmatose -s 20.4.3.219
cmatose: starting client
cmatose: connecting
receiving data transfers
sending replies
data transfers complete
test complete
return status 0
#
```

This server-client run is without PCP or VLAN because the IP address used does not belong to a VLAN interface. If you specify a VLAN IP address, then the traffic should go over VLAN.

3.3.1.7.5 Type Of Service (ToS)

3.3.1.7.5.1 Overview

The TOS field for rdma_cm sockets can be set using the rdma_set_option() API, just as it is set for regular sockets. If a TOS is not set, the default value (0) is used. Within the rdma_cm kernel driver, the TOS field is converted into an SL field. The conversion formula is as follows:

- SL = TOS >> 5 (e.g., take the 3 most significant bits of the TOS field)

In the hardware driver, the SL field is converted into PCP by the following formula:

- PCP = SL & 7 (take the 3 least significant bits of the TOS field)

SL affects the PCP only when the traffic goes over tagged VLAN frames.

3.3.1.7.5.2 DSCP

A new entry has been added to the RDMA-CM configfs that allows users to select default TOS for RDMA-CM QPs. This is useful for users that want to control the TOS field without changing their code. Other applications that set the TOS explicitly using the rdma_set_option API will continue to work as expected to override the configfs value.
For further information about DSCP marking, refer to HowTo Set Egress ToS/DSCP on RDMA CM QPs Community post.

3.3.1.7.6 RoCE LAG

RoCE LAG is a feature meant for mimicking Ethernet bonding for IB devices and is available for dual port cards only.
This feature is supported on kernel versions 4.9 and above.

RoCE LAG mode is entered when both Ethernet interfaces are configured as a bond in one of the following modes:

- active-backup (mode 1)
- balance-xor (mode 2)
- 802.3ad (LACP) (mode 4)

Any change of bonding configuration that negates one of the above rules (i.e, bonding mode is not 1, 2 or 4, or both Ethernet interfaces that belong to the same card are not the only slaves of the bond interface), will result in exiting RoCE LAG mode and the return to normal IB device per port configuration.
Once RoCE LAG is enabled, instead of having two IB devices; mlx5_0 and mlx5_1, there will be one device named mlx5_bond_0.
For information on how to configure RoCE LAG, refer to HowTo Configure RoCE over LAG (ConnectX-4/ConnectX-5/ConnectX-6) Community post.

3.3.1.7.7 Disabling RoCE

By default, RoCE is enabled on all mlx5 devices. When RoCE is enabled, all traffic to UDP port 4791 is treated as RoCE traffic by the device.

In case you are only interested in Ethernet (no RDMA) and wish to enable forwarding of traffic to this port, you can disable RoCE through sysfs:

```
echo <0|1> > /sys/devices/{pci-bus-address}/roce_enable
```

⚠ Once RoCE is disabled, only Ethernet traffic will be supported. Therefore, there will be no GID tables and only Raw Ethernet QPs will be supported.

The current RoCE state can be queried by sysfs:

```
cat /sys/devices/{pci-bus-address}/roce_enable
```

3.3.1.7.8 Enabling/Disabling RoCE on VMs via VFs

By default, when configuring VFs on the hypervisor, all VFs will be enabled with RoCE. This means they require more OS memory (from the VM). In case you are only interested in Ethernet (no RDMA) on the VM, and you wish to save the VM memory, you can disable RoCE on the VF from the hypervisor. In addition, by disabling RoCE, a VM can have the capability of utilizing the RoCE UDP port (4791) for standard UDP traffic.

For details on how to enable/disable RoCE on a VF, refer to HowTo Enable/Disable RoCE on VMs via VFs Community post.

3.3.1.7.9 Force DSCP

This feature enables setting a global traffic_class value for all RC QPs, or setting a specific traffic class based on several matching criteria.

Usage

- To set a single global traffic class to be applied to all QPs, write the desired global traffic_class value to /sys/class/infiniband/<dev>/tc/<port>/traffic_class.
  Note the following:
  - Negative values indicate that the feature is disabled. traffic_class value can be set using ibv_modify_qp()
  - Valid values range between 0 - 255

⚠ The ToS field is 8 bits, while the DSCP field is 6 bits. To set a DSCP value of X, you need to multiply this value by 4 (SHIFT 2). For example, to set DSCP value of 24, set the ToS bit to 96 (24x4=96).

- To set multiple traffic class values based on source and/or destination IPs, write the desired rule to /sys/class/infiniband/<dev>/tc/<port>/traffic_class. For example:

  ```
  echo "tclass=16,src_ip=1.1.1.2,dst_ip=1.1.1.0/24" > /sys/class/infiniband/mlx5_0/tc/1/traffic_class
  ```

  Note: Adding "tclass" prefix to tclass value is optional.

In the example above, traffic class 16 will be set to any QP with source IP 1.1.1.2 and destination IP 1.1.1.0/24.
Note that when setting a specific traffic class, the following rule precedence will apply:

- If a global traffic class value is set, it will be applied to all QPs

- If no global traffic class value is set, and there is a rule with matching source and destination IPs applicable to at least one QP, it will be applied
- Rules only with matching source and/or destination IPs have no defined precedence over other rules with matching source and/or destination IPs

Notes:

- A mask can be provided when using destination IPv4 addresses
- The rule precedence is not affected by the order in which rules are inserted
- Overlapping rules are entirely up to the administrator.
- "tclass=-1" will remove the rule from the database

3.3.1.7.10 Force Time to Live (TTL)

This feature enables setting a global TTL value for all RC QPs.
Write the desired TTL value to /sys/class/infiniband/<dev>/tc/<port>/ttl. Valid values range between 0 - 255

3.3.1.8 Flow Control

3.3.1.8.1 Priority Flow Control (PFC)

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

3.3.1.8.1.1 Configuring PFC on ConnectX-4 and above

1. Enable PFC on the desired priority:

   ```
   mlnx_qos -i <ethX> --pfc <0/1>,<0/1>,<0/1>,<0/1>,<0/1>,<0/1>,<0/1>,<0/1>
   ```

   Example (Priority=4):

   ```
   mlnx_qos -i eth1 --pfc 0,0,0,0,1,0,0,0
   ```

2. Create a VLAN interface:

   ```
   vconfig add <ethX> <VLAN_ID>
   ```

   Example (VLAN_ID=5):

   ```
   vconfig add eth1 5
   ```

3. Set egress mapping:
   a. For Ethernet traffic:

      ```
      vconfig set_egress_map <vlan_einterface> <skprio> <up>
      ```

      Example (skprio=3, up=5):

```
vconfig set_egress_map eth1.5 3 5
```

4. Create 8 Traffic Classes (TCs):

```
tc_wrap.py -i <interface>
```

5. Enable PFC on the switch.
   For information on how to enable PFC on your respective switch, please refer to Switch FC/PFC Configuration sections in the RDMA/RoCE Solutions Community page.

3.3.1.8.1.2 PFC Configuration Using LLDP DCBX

PFC Configuration on Hosts

PFC Auto-Configuration Using LLDP Tool in the OS

1. Start lldpad daemon on host.

```
lldpad -d Or
service lldpad start
```

2. Send lldpad packets to the switch.

```
lldptool set-lldp -i <ethX>  adminStatus=rxtx  ;
lldptool -T -i <ethX>   -V  sysName enableTx=yes;
lldptool -T -i <ethX>   -V  portDesc enableTx=yes ;
lldptool -T -i <ethX>   -V  sysDesc enableTx=yes
lldptool -T -i <ethX>   -V sysCap enableTx=yess
lldptool -T -i <ethX>   -V mngAddr enableTx=yess
lldptool -T -i <ethX>   -V PFC enableTx=yes;
lldptool -T -I <ethX>   -V CEE-DCBX enableTx=yes;
```

3. Set the PFC parameters.

   - For the CEE protocol, use dcbtool:

```
dcbtool sc <ethX> pfc pfcup:<xxxxxxxx>
```

   Example:

```
dcbtool sc eth6 pfc pfcup:01110001
```

   where:

[pfcup:xxxxxxx]	Enables/disables priority flow control. From left to right (priorities 0-7) - x can be equal to either 0 or 1. 1 indicates that the priority is configured to transmit priority pause.

   - For IEEE protocol, use lldptool:

```
lldptool -T -i <ethX> -V PFC enabled=x,x,x,x,x,x,x,x
```

   Example:

```
lldptool -T -i eth2 -V PFC enabled=1,2,4
```

   where:

enabled	Displays or sets the priorities with PFC enabled. The set attribute takes a comma-separated list of priorities to enable, or the string none to disable all priorities.

PFC Auto-Configuration Using LLDP in the Firmware (for mlx5 driver)

There are two ways to configure PFC and ETS on the server:

1. Local Configuration - Configuring each server manually.
2. Remote Configuration - Configuring PFC and ETS on the switch, after which the switch will pass the configuration to the server using LLDP DCBX TLVs.
   There are two ways to implement the remote configuration using mlx5 driver:
   a. Configuring the adapter firmware to enable DCBX.
   b. Configuring the host to enable DCBX.

For further information on how to auto-configure PFC using LLDP in the firmware, refer to the HowTo Auto-Config PFC and ETS on ConnectX-4 via LLDP DCBX Community post.

PFC Configuration on Switches

1. In order to enable DCBX, LLDP should first be enabled:

```
switch (config) # lldp
show lldp interfaces ethernet remote
```

2. Add DCBX to the list of supported TLVs per required interface.
   For IEEE DCBX:

```
switch (config) # interface 1/1
switch (config interface ethernet 1/1) # lldp tlv-select dcbx
```

   For CEE DCBX:

```
switch (config) # interface 1/1
switch (config interface ethernet 1/1) # lldp tlv-select dcbx-cee
```

3. [Optional] Application Priority can be configured on the switch, with the required ethertype and priority. For example, IP packet, priority 1:

```
switch (config) # dcb application-priority 0x8100 1
```

4. Make sure PFC is enabled on the host (for enabling PFC on the host, refer to PFC Configuration on Hosts section above). Once it is enabled, it will be passed in the LLDP TLVs.
5. Enable PFC with the desired priority on the Ethernet port.

```
dcb priority-flow-control enable force
dcb priority-flow-control priority <priority> enable
interface ethernet <port> dcb priority-flow-control mode on force
```

   Example - Enabling PFC with priority 3 on port 1/1:

```
dcb priority-flow-control enable force
dcb priority-flow-control priority 3 enable
interface ethernet 1/1 dcb priority-flow-control mode on force
```

Priority Counters

Several ingress and egress counters per priority are supported. Run ethtool -S to get the full list of port counters.

ConnectX-4 Counters

- Rx and Tx Counters:
  - Packets
  - Bytes
  - Octets
  - Frames
  - Pause
  - Pause frames
  - Pause Duration
  - Pause Transition

ConnectX-4 Example

```
# ethtool -S eth35 | grep prio4
     prio4_rx_octets: 62147780800
     prio4_rx_frames: 14885696
     prio4_tx_octets: 0
     prio4_tx_frames: 0
     prio4_rx_pause: 0
     prio4_rx_pause_duration: 0
     prio4_tx_pause: 26832
     prio4_tx_pause_duration: 14508
     prio4_rx_pause_transition: 0
```

Note: The Pause counters in ConnectX-4 are visible via ethtool only for priorities on which PFC is enabled.

3.3.1.8.1.3 PFC Storm Prevention

PFC storm prevention enables toggling between default and auto modes.
The stall prevention timeout is configured to 8 seconds by default. Auto mode sets the stall prevention timeout to be 100 msec.
The feature can be controlled using sysfs in the following directory: /sys/class/net/eth*/settings/pfc_stall_prevention

- To query the PFC stall prevention mode:

  ```
  cat /sys/class/net/eth*/settings/pfc_stall_prevention
  ```

  Example

  ```
  $ cat /sys/class/net/ens6/settings/pfc_stall_prevention
  default
  ```

- To configure the PFC stall prevention mode:

  ```
  Echo "auto"/"default" > /sys/class/net/eth*/settings/pfc_stall_prevention
  ```

The following two counters were added to the ethtool -S:

- `tx_Pause_storm_warning_events` - when the device is stalled for a period longer than a pre-configured watermark, the counter increases, allowing the debug utility an insight into current device status.
- `tx_pause_storm_error_events` - when the device is stalled for a period longer than a pre-configured timeout, the pause transmission is disabled, and the counter increase.

3.3.1.8.2 Dropless Receive Queue (RQ)

Dropless RQ feature enables the driver to notify the FW when SW receive queues are overloaded. This scenario takes place when the handling of SW receive queue is slower than the handling of the HW receive queues.
When this feature is enabled, a packet that is received while the receive queue is full will not be immediately dropped. The FW will accumulate these packets assuming posting of new WQEs will resume shortly. If received WQEs are not posted after a certain period of time, out_of_buffer counter will increase, indicating that the packet has been dropped.
This feature is disabled by default. In order to activate it, ensure that Flow Control feature is also enabled.

To enable the feature, run:

```
ethtool --set-priv-flags ens6 dropless_rq on
```

To get the feature state, run:

```
ethtool --show-priv-flags DEVNAME
```

Output example:

```
Private flags for DEVNAME:
rx_cqe_moder   : on
rx_cqe_compress: off
sniffer        : off
dropless_rq    : off
hw_lro         : off
```

To disable the feature, run:

```
ethtool --set-priv-flags ens6 dropless_rq off
```

3.3.1.9 Explicit Congestion Notification (ECN)

ECN is an extension to the IP protocol. It allows reliable communication by notifying all ends of communication when congestion occurs. This is done without dropping packets.
Please note that this feature requires all nodes in the path (nodes, routers etc) between the communicating nodes to support ECN to ensure reliable communication. ECN is marked as 2 bits in the traffic control IP header. This ECN implementation refers to RoCE v2.

3.3.1.9.1 Enabling ECN

To enable ECN on the hosts:

1. Enable ECN in sysfs.

```
/sys/class/net/<interface>/<protocol>/ecn_<protocol>_enable =1
```

2. Query the attribute.

```
cat /sys/class/net/<interface>/ecn/<protocol>/params/<requested attribute>
```

3. Modify the attribute.

```
echo <value> /sys/class/net/<interface>/ecn/<protocol>/params/<requested attribute>
```

ECN supports the following algorithms:

- r_roce_ecn_rp - Reaction point
- r_roce_ecn_np - Notification point

Each algorithm has a set of relevant parameters and statistics, which are defined per device, per port, per priority.

To query whether ECN is enabled per Priority X:

```
cat /sys/class/net/<interface>/ecn/<protocol>/enable/X
```

To read ECN configurable parameters:

```
cat /sys/class/net/<interface>/ecn/<protocol>/requested attributes
```

To enable ECN for each priority per protocol:

```
echo 1 > /sys/class/net/<interface>/ecn/<protocol>/enable/X
```

To modify ECN configurable parameters:

```
echo <value> > /sys/class/net/<interface>/ecn/<protocol>/requested attributes
```

where:

- X: priority {0..7}
- protocol: roce_rp / roce_np
- requested attributes: Next Slide for each protocol.

3.3.1.10 RSS Support

3.3.1.10.1 RSS Hash Function

The device has the ability to use XOR as the RSS distribution function, instead of the default Toplitz function.
The XOR function can be better distributed among driver's receive queues in a small number of streams, where it distributes each TCP/UDP stream to a different queue. MLNX_OFED MLNX_EN provides the following option to change the working RSS hash function from Toplitz to XOR, and vice-versa:

Through sysfs, located at: /sys/class/net/eth*/settings/hfunc.

To query the operational and supported hash functions:

```
cat /sys/class/net/eth*/settings/hfunc
```

Example:

```
cat /sys/class/net/eth2/settings/hfunc
Operational hfunc: toeplitz
Supported hfuncs: xor toeplitz
```

To change the operational hash function:

```
echo xor > /sys/class/net/eth*/settings/hfunc
```

3.3.1.10.1.1 RSS Verbs Support

Receive Side Scaling (RSS) technology allows spreading incoming traffic between different receive descriptor queues. Assigning each queue to different CPU cores allows better load balancing of the incoming traffic and improves performance.
This technology was extended to user space by the verbs layer and can be used for RAW ETH QP.

3.3.1.10.1.2 RSS Flow Steering

Steering rules classify incoming packets and deliver a specific traffic type (e.g. TCP/UDP, IP only) or a specific flow to "RX Hash" QP. "RX Hash" QP is responsible for spreading the traffic it handles between the Receive Work Queues using RX hash and Indirection Table. The Receive Work Queue can point to different CQs that can be associated with different CPU cores.

3.3.1.10.1.3 Verbs

The below verbs should be used to achieve this task in both control and data path. Details per verb should be referenced from its man page.

- ibv_create_wq, ibv_modify_wq, ibv_destory_wq
- ibv_create_rwq_ind_table, ibv_destroy_rwq_ind_table
- ibv_create_qp_ex with specific RX configuration to create the "RX hash" QP

3.3.1.11 Time-Stamping

3.3.1.11.1 Time-Stamping Service

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

3.3.1.11.1.1 Enabling Time-Stamping

Time-stamping is off by default and should be enabled before use.

To enable time-stamping for a socket:

Call setsockopt() with SO_TIMESTAMPING and with the following flags:

SOF_TIMESTAMPING_TX_HARDWARE:	try to obtain send time-stamp in hardware
SOF_TIMESTAMPING_TX_SOFTWARE:	if SOF_TIMESTAMPING_TX_HARDWARE is off or fails, then do it in software

SOF_TIMESTAMPING_RX_HARDWARE:	return the original, unmodified time-stamp as generated by the hardware
SOF_TIMESTAMPING_RX_SOFTWARE:	if SOF_TIMESTAMPING_RX_HARDWARE is off or fails, then do it in software
SOF_TIMESTAMPING_RAW_HARDWARE :	return original raw hardware time-stamp
SOF_TIMESTAMPING_SYS_HARDWARE:	return hardware time-stamp transformed into the system time base
SOF_TIMESTAMPING_SOFTWARE:	return system time-stamp generated in software
SOF_TIMESTAMPING_TX/RX	determine how time-stamps are generated
SOF_TIMESTAMPING_RAW/SYS	determine how they are reported

To enable time-stamping for a net dev*ice:*

Admin privileged user can enable/disable time stamping through calling ioctl (sock, SIOCSH-WTSTAMP, &ifreq) with the following values:

- Send side time sampling, enabled by ifreq.hwtstamp_config.tx_type when:

```
/* possible values for hwtstamp_config->tx_type */
enum hwtstamp_tx_types {
        /*
         * No outgoing packet will need hardware time stamping;
         * should a packet arrive which asks for it, no hardware
         * time stamping will be done.
        */
        HWTSTAMP_TX_OFF,

        /*
         * Enables hardware time stamping for outgoing packets;
         * the sender of the packet decides which are to be
         * time stamped by setting %SOF_TIMESTAMPING_TX_SOFTWARE
         * before sending the packet.
         */
        HWTSTAMP_TX_ON,
/*
         * Enables time stamping for outgoing packets just as
         * HWTSTAMP_TX_ON does, but also enables time stamp insertion
         * directly into Sync packets. In this case, transmitted Sync
         * packets will not received a time stamp via the socket error
         * queue.
         */
        HWTSTAMP_TX_ONESTEP_SYNC,
};
Note: for send side time stamping currently only HWTSTAMP_TX_OFF and
HWTSTAMP_TX_ON are supported.
```

- Receive side time sampling, enabled by ifreq.hwtstamp_config.rx_filter when:

```
/* possible values for hwtstamp_config->rx_filter */
enum hwtstamp_rx_filters {
        /* time stamp no incoming packet at all */
        HWTSTAMP_FILTER_NONE,

        /* time stamp any incoming packet */
        HWTSTAMP_FILTER_ALL,
      /* return value: time stamp all packets requested plus some others */
        HWTSTAMP_FILTER_SOME,

        /* PTP v1, UDP, any kind of event packet */
        HWTSTAMP_FILTER_PTP_V1_L4_EVENT,
        /* PTP v1, UDP, Sync packet */
        HWTSTAMP_FILTER_PTP_V1_L4_SYNC,
        /* PTP v1, UDP, Delay_req packet */
        HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ,
        /* PTP v2, UDP, any kind of event packet */
        HWTSTAMP_FILTER_PTP_V2_L4_EVENT,
        /* PTP v2, UDP, Sync packet */
        HWTSTAMP_FILTER_PTP_V2_L4_SYNC,
        /* PTP v2, UDP, Delay_req packet */
        HWTSTAMP_FILTER_PTP_V2_L4_DELAY_REQ,
```

```
        /* 802.AS1, Ethernet, any kind of event packet */
        HWTSTAMP_FILTER_PTP_V2_L2_EVENT,
        /* 802.AS1, Ethernet, Sync packet */
        HWTSTAMP_FILTER_PTP_V2_L2_SYNC,
        /* 802.AS1, Ethernet, Delay_req packet */
        HWTSTAMP_FILTER_PTP_V2_L2_DELAY_REQ,

        /* PTP v2/802.AS1, any layer, any kind of event packet */
        HWTSTAMP_FILTER_PTP_V2_EVENT,
        /* PTP v2/802.AS1, any layer, Sync packet */
        HWTSTAMP_FILTER_PTP_V2_SYNC,
        /* PTP v2/802.AS1, any layer, Delay_req packet */
        HWTSTAMP_FILTER_PTP_V2_DELAY_REQ,
};
Note: for receive side time stamping currently only HWTSTAMP_FILTER_NONE and
HWTSTAMP_FILTER_ALL are supported.
```

3.3.1.11.1.2 Getting Time-Stamping

Once time stamping is enabled time stamp is placed in the socket Ancillary data. recvmsg() can be used to get this control message for regular incoming packets. For send time stamps the outgoing packet is looped back to the socket's error queue with the send time-stamp(s) attached. It can be received with recvmsg (flags=MSG_ERRQUEUE). The call returns the original outgoing packet data including all headers prepended down to and including the link layer, the scm_time-stamping control message and a sock_extended_err control message with ee_errno==ENOMSG and ee_origin==SO_EE_ORIGIN_TIMESTAMPING. A socket with such a pending bounced packet is ready for reading as far as select() is concerned. If the outgoing packet has to be fragmented, then only the first fragment is time stamped and returned to the sending socket.

⚠ When time-stamping is enabled, VLAN stripping is disabled. For more info please refer to Documentation/networking/timestamping.txt in kernel.org.

⚠ On ConnectX-4 and above adapter cards, when time-stamping is enabled, RX CQE compression is disabled (features are mutually exclusive).

3.3.1.11.1.3 Time Stamping Capabilities via ethtool

To display Time Stamping capabilities via ethtool:

Show Time Stamping capabilities:

```
ethtool -T eth<x>
```

Example:

```
ethtool -T eth0
Time stamping parameters for p2p1:
Capabilities:
                                hardware-transmit           (SOF_TIMESTAMPING_TX_HARDWARE)
                                software-transmit           (SOF_TIMESTAMPING_TX_SOFTWARE)
                                hardware-receive            (SOF_TIMESTAMPING_RX_HARDWARE)
                                software-receive            (SOF_TIMESTAMPING_RX_SOFTWARE)
                                software-system-clock       (SOF_TIMESTAMPING_SOFTWARE)
                                hardware-raw-clock          (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 1
Hardware Transmit Timestamp Modes:
off                               (HWTSTAMP_TX_OFF)
on                                (HWTSTAMP_TX_ON)

Hardware Receive Filter Modes:
```

```
none                                (HWTSTAMP_FILTER_NONE)
all                                 (HWTSTAMP_FILTER_ALL)
```

For more details on PTP Hardware Clock, please refer to: https://www.kernel.org/doc/Documentation/ptp/ptp.txt

3.3.1.11.1.4 Steering PTP Traffic to Single RX Ring

As a result of Receive Side Steering (RSS) PTP traffic coming to UDP ports 319 and 320, it may reach the user space application in an out of order manner. In order to prevent this, PTP traffic needs to be steered to single RX ring using ethtool.

Example:

```
# ethtool -u ens7
8 RX rings available
Total 0 rules
# ethtool -U ens7 flow-type udp4 dst-port 319 action 0 loc 1
# ethtool -U ens7 flow-type udp4 dst-port 320 action 0 loc 0
# ethtool -u ens7
8 RX rings available
Total 2 rules
Filter: 0
Rule Type: UDP over IPv4
Src IP addr: 0.0.0.0 mask: 255.255.255.255
Dest IP addr: 0.0.0.0 mask: 255.255.255.255
TOS: 0x0 mask: 0xff
Src port: 0 mask: 0xffff
Dest port: 320 mask: 0x0
Action: Direct to queue 0
Filter: 1
Rule Type: UDP over IPv4
Src IP addr: 0.0.0.0 mask: 255.255.255.255
Dest IP addr: 0.0.0.0 mask: 255.255.255.255
TOS: 0x0 mask: 0xff
Src port: 0 mask: 0xffff
Dest port: 319 mask: 0x0
Action: Direct to queue 0
```

3.3.1.11.1.5 Tx Port Time-Stamping

Transmitted packet time-stamping accuracy can be improved when using a timestamp generated at the port level instead of a timestamp generated upon CQE creation. Tx port time-stamping better reflects the actual time of a packet's transmission.

Normal Send queues (SQs) are open with CQE time-stamp support. When this feature is enabled, the driver is expected to open extra Tx port time-stamped SQ per traffic class (TC).

The stream must meet the following conditions in order to be transmitted through a Tx port time-stamped SQ.

1. SKBTX_HW_TSTAMP flag was set at tx_flag (SO_TIMESTAMPING was set via setsockopt() or similarly)
2. Packet type is:
    a. Non-IP, with EtherType of PTP over IEEE 802.3 (0x88f7)
       or
    b. UDP over IPv4/IPv6

This feature is disabled by default in order to avoid extra SQ memory allocations. The feature can be enabled or disabled using the following command.

```
ethtool --set-priv-flags <ifs-name> tx_port_ts on / off
```

3.3.1.11.1.6 PTP Cyc2time Hardware Translation Offload

This feature is supported on ConnectX-6 Dx and above adapter cards only.

Overview

Device timestamp can be in one of two modes: real time or free running internal time.
In free running internal time mode, the device clock is not editable in any way. Driver and/or user space must adjust it to the real-time nanosecond values.
In real time mode, the hardware clock device can be adjusted and can provide timestamps which are already translated into real-time nanoseconds.

Both modes are global per device. Once a mode is set, all clock-related features (such as PPS, CQE TS, PCIe bar, etc) will work with the chosen clock mode only.

Free running internal time is the default mode configured in the hardware. The driver will modify the hardware real time clock based on PTP daemon clock adjustments.

Only physical functions are allowed to modify the hardware real-time clock, so PTP daemon adjustments from VFs will be treated as NOP. In case more than one physical function tries to modify the hardware real-time clock, the device will select one of the functions as its designated clock provider. All other input will also be treated as a NOP. The designated clock provide can be replaced by the device if no new adjustments have been received from the current provider after some period.

Timestamp Format

CQE hardware timestamp format for ConnectX-6 Dx and ConnectX-6 Lx NICs is 64 bit, as follows.

{32bit sec, 32 bit nsec}

Configuration

In order to enable the feature, set REAL_TIME_CLOCK_ENABLE in NV_CONFIG via mlxconfig and restart the driver.

Limitations

- Administrator must restart the driver and perform a FW reset for the configuration to take effect. Otherwise, mismatch between HW and driver timestamp mode might occur.
- Once real time mode is activated on a given device (see configuration section), version 5.3 or newer must run on all device functions. Any older driver running on a device function at this configuration will fail to open any traffic queues (RDMA or ETH), hence becoming dysfunctional.
- In real time mode, all device functions must be PTP-synchronized by a single clock domain—do not use multiple GMs for different functions on the same device.
- Regarding hardware clock ownership, the hardware is configured only from a single elected function; other function settings are ignored by the device. There is no indication as to which function is the hardware-clock's owner. After an internal timeout without modifying the

hardware clock, a function losses the hardware-clock's ownership and is open to be grasped by any of the functions.
- All PFs/VFs within the same device must sync to the same 1588 master clock. If multiple masters are used, the device will use a single elected function. This might lead to wrong clock representation by device, wrong 1588 TLVs and hiccups on replacement of elected function.
- This feature is supported on ConnectX-6 Dx and above adapter cards only.

3.3.1.11.2 RoCE Time-Stamping

RoCE Time-Stamping allows you to stamp packets when they are sent to the wire/received from the wire. The time-stamp is given in raw hardware cycles but could be easily converted into hardware referenced nanoseconds based time. Additionally, it enables you to query the hardware for the hardware time, thus stamp other application's event and compare time.

3.3.1.11.2.1 Query Capabilities

Time-stamping is available if and only the hardware reports it is capable of reporting it. To verify whether RoCE Time-Stamping is available, run ibv_query_device_ex.

For further information, please see ibv_query_device_ex manual page.

3.3.1.11.2.2 Creating a Time-Stamping Completion Queue

To get time stamps, a suitable extended Completion Queue (CQ) must be created via a special call to ibv_create_cq_ex verb.

For further information, please see ibv_create_cq_ex manual page.

> ⚠ Time Stamping in not available when CQE zipping is used.

3.3.1.11.2.3 Querying the Hardware Time

Querying the hardware for time is done via the ibv_query_rt_values_ex verb. For example:

For further information, please see ibv_query_rt_values_ex manual page.

3.3.1.11.3 One Pulse Per Second (1PPS)

1PPS is a time synchronization feature that allows the adapter to be able to send or receive 1 pulse per second on a dedicated pin on the adapter card using an SMA connector (SubMiniature version A). Only one pin is supported and could be configured as 1PPS in or 1PPS out.
For further information, refer to HowTo Test 1PPS on NVIDIA Adapters Community post.

3.3.1.12 Flow Steering

Flow steering is a new model which steers network flows based on flow specifications to specific QPs. Those flows can be either unicast or multicast network flows. In order to maintain flexibility, domains and priorities are used. Flow steering uses a methodology of flow attribute, which is a combination of L2-L4 flow specifications, a destination QP and a priority. Flow steering rules may be

inserted either by using ethtool or by using InfiniBand verbs. The verbs abstraction uses different terminology from the flow attribute (ibv_flow_attr), defined by a combination of specifications (struct ibv_flow_spec_*).

3.3.1.12.1 Flow Steering Support

All flow steering features are enabled in the supported adapter cards. Flow Steering support in InfiniBand is determined according to the MANAGED_FLOW_STEERING flag.

3.3.1.12.2 Flow Domains and Priorities

Flow steering defines the concept of domain and priority. Each domain represents a user agent that can attach a flow. The domains are prioritized. A higher priority domain will always supersede a lower priority domain when their flow specifications overlap. Setting a lower priority value will result in a higher priority.
In addition to the domain, there is a priority within each of the domains. Each domain can have at most 2^12 priorities in accordance with its needs.
The following are the domains at a descending order of priority:

- User Verbs allows a user application QP to be attached to a specified flow when using `ibv_create_flow` and `ibv_destroy_flow` verbs
  - `ibv_create_flow`

    ```
    struct ibv_flow *ibv_create_flow(struct ibv_qp *qp, struct ibv_flow_attr
    *flow)
    ```

    Input parameters:
    - `struct ibv_qp` - the attached QP.
    - `struct ibv_flow_attr` - attaches the QP to the flow specified. The flow contains mandatory control parameters and optional L2, L3 and L4 headers. The optional headers are detected by setting the size and `num_of_specs` fields: `struct ibv_flow_attr` can be followed by the optional flow headers structs:

      ```
      struct ibv_flow_spec_eth
      struct ibv_flow_spec_ipv4
      struct ibv_flow_spec_tcp_udp
      struct ibv_flow_spec_ipv6
      ```

      For further information, please refer to the `ibv_create_flow` man page.
  - `ibv_destroy_flow`

    ```
    int ibv_destroy_flow(struct ibv_flow *flow_id)
    ```

    Input parameters:
    `ibv_destroy_flow` requires struct `ibv_low` which is the return value of `ibv_create_flow` in case of success.
    Output parameters:
    Returns 0 on success, or the value of errno on failure.
    For further information, please refer to the `ibv_destroy_flow` man page.

3.3.1.12.3 Ethtool

Ethtool domain is used to attach an RX ring, specifically its QP to a specified flow. Please refer to the most recent ethtool man page for all the ways to specify a flow.

Examples:

- ethtool -U eth5 flow-type ether dst 00:11:22:33:44:55 loc 5 action 2
  All packets that contain the above destination MAC address are to be steered into rx-ring 2 (its underlying QP), with priority 5 (within the ethtool domain)
- ethtool -U eth5 flow-type tcp4 src-ip 1.2.3.4 dst-port 8888 loc 5 action 2
  All packets that contain the above destination IP address and source port are to be steered into rx- ring 2. When destination MAC is not given, the user's destination MAC is filled automatically.
- ethtool -U eth5 flow-type ether dst 00:11:22:33:44:55 vlan 45 m 0xf000 loc 5 action 2
  All packets that contain the above destination MAC address and specific VLAN are steered into ring 2. Please pay attention to the VLAN's mask 0xf000. It is required in order to add such a rule.
- ethtool -u eth5
  Shows all of ethtool's steering rule

When configuring two rules with the same priority, the second rule will overwrite the first one, so this ethtool interface is effectively a table. Inserting Flow Steering rules in the kernel requires support from both the ethtool in the user space and in kernel (v2.6.28).

3.3.1.12.4 Accelerated Receive Flow Steering (aRFS)

Receive Flow Steering (RFS) and Accelerated Receive Flow Steering (aRFS) are kernel features currently available in most distributions. For RFS, packets are forwarded based on the location of the application consuming the packet. aRFS boosts the speed of RFS by adding support for the hardware. By usingaRFS(unlike RFS), the packets are directed to a CPU that is local to the thread running the application.

aRFSis an in-kernel-logic responsible for load balancing between CPUs by attaching flows to CPUs that are used by flow's owner applications. This domain allows the aRFS mechanism to use the flow steering infrastructure to support the aRFS logic by implementing the ndo_rx_flow_steer, which, in turn, calls the underlying flow steering mechanism with the aRFS domain.

To configure RFS:

Configure the RFS flow table entries (globally and per core).
Note: The functionality remains disabled until explicitly configured (by default it is 0).

- The number of entries in the global flow table is set as follows:

  /proc/sys/net/core/rps_sock_flow_entries

- The number of entries in the per-queue flow table are set as follows:

/sys/class/net/<dev>/queues/rx-<n>/rps_flow_cnt

Example:

```
# echo 32768 > /proc/sys/net/core/rps_sock_flow_entries
# NUM_CHANNELS=`ethtool -l ens6 | grep "Combined:" | tail -1 | awk '{print $2}'`
# for f in `seq 0 $((NUM_CHANNELS-1))`; do echo 32768 > /sys/class/net/ens6/queues/rx-$f/rps_flow_cnt; done
```

To Configure aRFS:

The aRFS feature requires explicit configuration in order to enable it. Enabling the aRFS requires enabling the 'ntuple' flag via the ethtool.
For example, to enable ntuple for eth0, run:

```
ethtool -K eth0 ntuple on
```

aRFS requires the kernel to be compiled with the `CONFIG_RFS_ACCEL` option. This option is available in kernels 2.6.39 and above. Furthermore, aRFS requires Device Managed Flow Steering support.

RFS cannot function if LRO is enabled. LRO can be disabled via ethtool.

3.3.1.12.5 Flow Steering Dump Tool

The mlx_fs_dump is a python tool that prints the steering rules in a readable manner. Python v2.7 or above, as well as pip, anytree and termcolor libraries are required to be installed on the host.

Running example:

```
./ofed_scripts/utils/mlx_fs_dump -d /dev/mst/mt4115_pciconf0
FT: 9 (level: 0x18, type: NIC_RX)
+-- FG: 0x15 (MISC)
    |-- FTE: 0x0 (FWD) to (TIR:0x7e) out.ethtype:IPv4 out.ip_prot:UDP out.udp_dport:0x140
    +-- FTE: 0x1 (FWD) to (TIR:0x7e) out.ethtype:IPv4 out.ip_prot:UDP out.udp_dport:0x13f
...
```

For further information on the mlx_fs_dump tool, please refer to mlx_fs_dump Community post.

3.3.1.13 Wake-on-LAN (WoL)

Wake-on-LAN (WoL) is a technology that allows a network professional to remotely power on a computer or to wake it up from sleep mode.

- To enable WoL:

```
ethtool -s <interface> wol g
```

- To get WoL:

```
ethtool <interface> | grep Wake-on Wake-on: g
```

Where:
"`g`" is the magic packet activity.

3.3.1.14 Hardware Accelerated 802.1ad VLAN (Q-in-Q Tunneling)

Q-in-Q tunneling allows the user to create a Layer 2 Ethernet connection between two servers. The user can segregate a different VLAN traffic on a link or bundle different VLANs into a single VLAN. Q-in-Q tunneling adds a service VLAN tag before the user's 802.1Q VLAN tags.
For Q-in-Q support in virtualized environments (SR-IOV), please refer to "Q-in-Q Encapsulation per VF in Linux (VST)".

To enable device support for accelerated 802.1ad VLAN:

1. Turn on the new ethtool private flag "phv-bit" (disabled by default).

```
$ ethtool --set-priv-flags eth1 phv-bit on
```

Enabling this flag sets the phv_en port capability.

2. Change the interface device features by turning on the ethtool device feature "tx-vlan- stag-hw-insert" (disabled by default).

```
$ ethtool -K eth1 tx-vlan-stag-hw-insert on
```

Once the private flag and the ethtool device feature are set, the device will be ready for 802.1ad VLAN acceleration.

The "phv-bit" private flag setting is available for the Physical Function (PF) only. The Virtual Function (VF) can use the VLAN acceleration by setting the "tx-vlan-stag-hw-insert" parameter only if the private flag "phv-bit" is enabled by the PF. If the PF enables/disables the "phv-bit" flag after the VF driver is up, the configuration will take place only after the VF driver is restarted.

3.3.1.15 VLAN Stripping in Linux Verbs

This capability is now accessible from userspace using the verbs.

VLAN stripping adds access to the device's ability to offload the Customer VLAN (cVLAN) header stripping from an incoming packet, thus achieving acceleration of VLAN handing in receive flow.

It is configured per WQ option. You can either enable it upon creation or modify it later using the appropriate verbs (`ibv_create_wq` / `ibv_modify_wq`).

3.3.1.16 Offloaded Traffic Sniffer

> To be able to activate this feature, make sure libpcap library v1.9 or above is installed on your setup.
>
> To download libpcap, please visit https://www.tcpdump.org/.

Offloaded Traffic Sniffer allows bypass kernel traffic (such as RoCE, VMA, and DPDK) to be captured by existing packet analyzer, such as tcpdump.

To capture the interface's bypass kernel traffic, run tcpdump on the RDMA device.

For examples on how to dump RDMA traffic using the Inbox tcpdump tool for ConnectX-4 adapter cards and above, click here.

> Note that enabling Offloaded Traffic Sniffer can cause bypass kernel traffic speed degradation.

> In case you do not wish to install libpcap on your setup, you can use docker to run the tcpdump. For further information, please see https://hub.docker.com/r/mellanox/tcpdump-rdma.

3.3.1.17 Dump Configuration

This feature helps dumping driver and firmware configuration using ethtool. It creates a backup of the configuration files into a specified dump file.

3.3.1.17.1 Dump Parameters (Bitmap Flag)

The following bitmap parameters are used to set the type of dump:

Bitmap Parameters

Value	Description
1	MST dump
2	Ring dump (Software context information for SQs, EQs, RQs, CQs)
3	MST dump + Ring dump (1+2)
4	Clear this parameter

3.3.1.17.2 Configuration

In order to configure this feature, follow the steps below:

1. Set the dump bitmap parameter by running -W (uppercase) with the desired bitmap parameter value (see Bitmap Parameters table above). In the following example, the bitmap parameter value is 3.

```
ethtool -W ens1f0 3
```

2. Dump the file by running -w (lowercase) with the desired configuration file name.

```
ethtool -w ens1f0 data /tmp/dump.bin
```

3. [Optional] To get the bitmap parameter value, version and size of the dump, run the command above without the file name.

```
ethtool -w ens1f0
flag: 3, version: 1, length: 4312
```

4. To open the dump file, run:

```
mlnx_dump_parser -f /tmp/dump.bin -m mst_dump_demo.txt -r ring_dump_demo.txt
Version: 1 Flag: 3 Number of blocks: 123 Length 327584
MCION module number: 0 status: | present |
DRIVER VERSION: 1-23 (03 Mar 2015)
DEVICE NAME 0000:81:00.0:ens1f0
Parsing Complete!
```

where:

-f	For the file to be parsed (the file that was just created)
-m	For the mst dump file
-r	For the ring dump file

For further information, refer to HowTo Dump Driver Configuration (via ethtool) Community post.
Output:

```
# mlnx_dump_parser -f /tmp/dump.bin -m mst_dump_demo.txt -r ring_dump_demo.txt
Version: 1 Flag: 3 Number of blocks: 123 Length 327584
MCION module number: 0 status: | present |
DRIVER VERSION: 1-23 (03 Mar 2015)
DEVICE NAME 0000:81:00.0:ens1f0
Parsing Complete!
```

5. Open the files.
    a. The MST dump file will look as follows. In order to analyze it, contact NVIDIA Support.

```
cat mst_dump_demo.txt
0x00000000 0x01002000
0x00000004 0x00000000
0x00000008 0x00000000
0x0000000c 0x00000000
0x00000010 0x00000000
0x00000014 0x00000000
0x00000018 0x00000000
...
```

    b. The Ring dump file can help developers debug ring-related issues, and it looks as follows:

```
# cat ring_dump_demo.txt
SQ TYPE: 3, WQN: 102, PI: 0, CI: 0, STRIDE: 6, SIZE: 1024...
SQ TYPE: 3, WQN: 102, PI: 0, CI: 0, STRIDE: 6, SIZE: 1024, WQE_NUM: 65536, GROUP_IP: 0
CQ TYPE: 5, WQN: 20, PI: 0, CI: 0, STRIDE: 6, SIZE: 1024, WQE_NUM: 1024, GROUP_IP: 0
```

```
RQ TYPE: 4, WQN: 103, PI: 15, CI: 0, STRIDE: 5, SIZE: 16, WQE_NUM: 512, GROUP_IP: 0
CQ TYPE: 5, WQN: 21, PI: 0, CI: 0, STRIDE: 6, SIZE: 16384, WQE_NUM: 16384, GROUP_IP: 0
EQ TYPE: 6, CI: 1, SIZE: 0, IRQN: 109, EQN: 19, NENT: 2048, MASK: 0, INDEX: 0, GROUP_ID: 0
SQ TYPE: 3, WQN: 106, PI: 0, CI: 0, STRIDE: 6, SIZE: 1024, WQE_NUM: 65536, GROUP_IP: 1
CQ TYPE: 5, WQN: 23, PI: 0, CI: 0, STRIDE: 6, SIZE: 1024, WQE_NUM: 1024, GROUP_IP: 1
RQ TYPE: 4, WQN: 107, PI: 15, CI: 0, STRIDE: 5, SIZE: 16, WQE_NUM: 512, GROUP_IP: 1
CQ TYPE: 5, WQN: 24, PI: 0, CI: 0, STRIDE: 6, SIZE: 16384, WQE_NUM: 16384, GROUP_IP: 1
EQ TYPE: 6, CI: 1, SIZE: 0, IRQN: 110, EQN: 20, NENT: 2048, MASK: 0, INDEX: 1, GROUP_ID: 1
SQ TYPE: 3, WQN: 110, PI: 0, CI: 0, STRIDE: 6, SIZE: 1024, WQE_NUM: 65536, GROUP_IP: 2
CQ TYPE: 5, WQN: 26, PI: 0, CI: 0, STRIDE: 6, SIZE: 1024, WQE_NUM: 1024, GROUP_IP: 2
RQ TYPE: 4, WQN: 111, PI: 15, CI: 0, STRIDE: 5, SIZE: 16, WQE_NUM: 512, GROUP_IP: 2
CQ TYPE: 5, WQN: 27, PI: 0, CI: 0, STRIDE: 6, SIZE: 16384, WQE_NUM: 16384, GROUP_IP: 2
...
```

3.3.1.18 Local Loopback Disable

Local Loopback Disable feature allows users to force the disablement of local loopback on the virtual port (vport). This disables both unicast and multicast loopback in the hardware.

To enable Local Loopback Disable, run the following command:

```
echo 1 > /sys/class/net/<ifname>/settings/force_local_lb_disable"
```

To disable Local Loopback Disable, run the following command:

```
echo 0 > /sys/class/net/<ifname>/settings/force_local_lb_disable"
```

When turned off, the driver configures the loopback mode according to its own logic.

3.3.1.19 Kernel Transport Layer Security (kTLS) Offloads

This feature is supported on ConnectX-6 Dx crypto cards only.

3.3.1.19.1 Overview

Transport Layer Security (TLS) is a widely-deployed protocol used for securing TCP connections on the Internet. TLS is also a required feature for HTTP/2, the latest web standard. Kernel implementation of TLS (kTLS) provides new opportunities for offloading the protocol into the hardware.

TLS data-path offload allows the NIC to accelerate encryption, decryption and authentication of AES-GCM. TLS offload handles data as it goes through the device without storing any data, but only updating context. If the packet cannot be encrypted/decrypted by the device, then a software fallback handles the packet.

3.3.1.19.2 Establishing a kTLS Connection

To avoid unnecessary complexity in the kernel, the TLS handshake is kept in the user space. A full TLS connection using the socket is done using the following scheme:

1. Call `connect()` or `accept()` on a standard TCP file descriptor.
2. Use a user space TLS library to complete a handshake.
3. Create a new KTLS socket file descriptor.
4. Extract the TLS Initialization Vectors (IVs), session keys, and sequence IDs from the TLS library. Use the `setsockopt` function on the kTLS file descriptor (FD) to pass them to the kernel.
5. Use standard `read()`, `write()`, `sendfile()` and `splice()` system calls on the KTLS FD.

Drivers can offer Tx and Rx packet encryption/decryption offload from the kernel into the NIC hardware. Upon receipt of a non-data TLS message (a control message), the kTLS socket returns an error, and the message is left on the original TCP socket instead. The kTLS socket is automatically unattached. Transfer of control back to the original encrypted FD is done by calling `getsockopt` to receive the current sequence numbers, and inserting them into the TLS library.

3.3.1.19.3 Kernel Support

For support in the kernel, make sure the following flags are set as follows.

- CONFIG_TLS=y
- CONFIG_TLS_DEVICE=y | m

> ⚠ For kTLS Tx device offloads with OFED drivers, kernel TLS module (kernel/net/tls) must be aligned to kernel v5.3 and above.
>
> For kTLS Rx device offloads with OFED drivers, kernel TLS module (kernel/net/tls) must be aligned to kernel v5.9 and above.

3.3.1.19.4 Configuring kTLS Offloads

➤ *To enable kTLS Tx offload, run:*

```
ethtool -K <ifs> tls-hw-tx-offload on
```

➤ *To enable kTLS Rx offload, run:*

```
ethtool -K <ifs> tls-hw-rx-offload on
```

For further information on TLS offloads, please visit the following kernel documentation:

- https://www.kernel.org/doc/html/latest/networking/tls-offload.html
- https://www.kernel.org/doc/html/latest/networking/tls.html#kernel-tls

3.3.1.19.5 OpenSSL with kTLS Offload

OpenSSL version 3.0.0 or above is required to support kTLS TX/RX offloads.

Supported OpenSSL version is available to download from distro packages, or can be downloaded and compiled from the OpenSSL github.

3.3.1.20 IPsec Crypto Offload

This feature is supported on crypto-enabled products of BlueField-2 DPUs, and ConnectX-6 Dx and ConnectX-7 adapters (but not of ConnectX-6 or ConnectX-6 Lx).

Newer/future crypto-enabled DPU and adapter product generations should also support the feature, unless explicitly stated in their documentation.

For NVIDIA BlueField-2 DPUs and ConnectX-6 Dx adapters Only: If your target application will utilize bandwidth of100Gb/s or higher, where a substantial part of the bandwidth will be allocated for IPsec traffic, please refer to the NVIDIA BlueField-2 DPUs Product Release Notes or NVIDIA ConnectX-6 Dx Adapters Product Release Notes document to learn about a potential bandwidth limitation. To access the relevant product release notes, please contact your NVIDIA sales representative.

3.3.1.20.1 Overview and Configuration

IPsec crypto offload feature, also known as IPsec inline offload or IPsec aware offload feature enables the user to offload IPsec crypto encryption and decryption operations to the hardware.

Note that the hardware implementation only supports AES-GCM encryption scheme.

To enable the feature, support in both kernel and adapter firmware is required.

- For support in the kernel, make sure the following flags are set as follows.

```
CONFIG_XFRM_OFFLOAD=y
CONFIG_INET_ESP_OFFLOAD=m
CONFIG_INET6_ESP_OFFLOAD=m
```

  Note: These flags are enabled by default in RedHat 8 and Ubuntu 18.04.

- For support in the firmware, make sure the below string is found in the dmesg.

```
mlx5e: IPSec ESP acceleration enabled
```

3.3.1.20.2 Configuring Security Associations for IPsec Offloads

To program the inline offload security associations (SA), add the option "offload dev <netdev interface> dir out/in" in the "ip xfrm state" command for transmitting and receiving SA.

Transmit inline offload SA xfrm command example:

```
sudo ip xfrm state add src 192.168.1.64/24 dst 192.168.1.65/24 proto esp spi 0x46dc6204 reqid 0x46dc6204 mode
transport aead 'rfc4106(gcm(aes))' 0x60bd6c3eafba371a46411830fd56c53af93883261ed1fb26767820ff493f43ba35b0dcca 128
 offload dev p4p1 dir out sel src 192.168.1.64 dst 192.168.1.65
```

Receive inline offload SA xfrm command example:

```
sudo ip xfrm state add src 192.168.1.65/24 dst 192.168.1.64/24 proto esp spi 0xaea0846c reqid 0xaea0846c mode
transport aead 'rfc4106(gcm(aes))' 0x81d5c3167c912c1dd50dab0cb4b6d815b6ace8844304db362215a258cd19deda8f89deda 128
 offload dev p4p1 dir in sel src 192.168.1.65 dst 192.168.1.64
```

3.3.1.20.2.1 Setting xfrm Policies Example

First server:

```
+ sudo ip xfrm state add src 192.168.1.64/24 dst 192.168.1.65/24 proto esp spi 0x28f39549 reqid 0x28f39549 mode
transport aead 'rfc4106(gcm(aes))' 0x492e8ffe718a95a00c1893ea61afc64997f4732848ccfe6ea07db483175cb18de9ae411a 128
 offload dev enp4s0 dir out sel src 192.168.1.64 dst 192.168.1.65
+ sudo ip xfrm state add src 192.168.1.65/24 dst 192.168.1.64/24 proto esp spi 0x622a73b4 reqid 0x622a73b4 mode
transport aead 'rfc4106(gcm(aes))' 0x093bfee2212802d626716815f862da31bcc7d9c44cfe3ab8049e7604b2feb1254869d25b 128
 offload dev enp4s0 dir in sel src 192.168.1.65 dst 192.168.1.64
+ sudo ip xfrm policy add src 192.168.1.64 dst 192.168.1.65 dir out tmpl src 192.168.1.64/24 dst 192.168.1.65/24
 proto esp reqid 0x28f39549 mode transport
+ sudo ip xfrm policy add src 192.168.1.65 dst 192.168.1.64 dir in tmpl src 192.168.1.65/24 dst 192.168.1.64/24
 proto esp reqid 0x622a73b4 mode transport
+ sudo ip xfrm policy add src 192.168.1.65 dst 192.168.1.64 dir fwd tmpl src 192.168.1.65/24 dst 192.168.1.64/24
 proto esp reqid 0x622a73b4 mode transport
```

Second server:

```
+ ssh -A -t root@l-csi-0921d /bin/bash
+ set -e
+ '[' 0 == 1 ']'
+ sudo ip xfrm state add src 192.168.1.64/24 dst 192.168.1.65/24 proto esp spi 0x28f39549 reqid 0x28f39549 mode
transport aead 'rfc4106(gcm(aes))' 0x492e8ffe718a95a00c1893ea61afc64997f4732848ccfe6ea07db483175cb18de9ae411a 128
 offload dev enp4s0 dir in sel src 192.168.1.64 dst 192.168.1.65
+ sudo ip xfrm state add src 192.168.1.65/24 dst 192.168.1.64/24 proto esp spi 0x622a73b4 reqid 0x622a73b4 mode
transport aead 'rfc4106(gcm(aes))' 0x093bfee2212802d626716815f862da31bcc7d9c44cfe3ab8049e7604b2feb1254869d25b 128
 offload dev enp4s0 dir out sel src 192.168.1.65 dst 192.168.1.64
+ sudo ip xfrm policy add src 192.168.1.65 dst 192.168.1.64 dir out tmpl src 192.168.1.65/24 dst 192.168.1.64/24
 proto esp reqid 0x622a73b4 mode transport
+ sudo ip xfrm policy add src 192.168.1.64 dst 192.168.1.65 dir in tmpl src 192.168.1.64/24 dst 192.168.1.65/24
 proto esp reqid 0x28f39549 mode transport
+ sudo ip xfrm policy add src 192.168.1.64 dst 192.168.1.65 dir fwd tmpl src 192.168.1.64/24 dst 192.168.1.65/24
 proto esp reqid 0x28f39549 mode transport
+ echo 'IPSec tunnel configured successfully'
```

3.3.1.21 IPsec Full Offload

This feature is supported on crypto-enabled products of BlueField-2 DPUs, as well as on ConnectX-6 Dx, ConnectX-6 Lx and ConnectX-7 adapter cards. Note that it is not supported on ConnectX-6 cards.

Newer/future crypto-enabled DPU and adapter product generations should also support this feature, unless explicitly stated otherwise in their documentation.

When using NVIDIA® BlueField®-2 DPUs and NVIDIA® ConnectX®-6 Dx adapters only: If your target application utilizes 100Gb/s or a higher bandwidth, where a substantial part of the bandwidth is allocated for IPsec traffic, please refer to the relevant DPU or adapter card Product Release Notes to learn about a potential bandwidth limitation. To access the Release Notes, visit https://docs.nvidia.com/networking/, or contact your NVIDIA sales representative.

ConnectX-6 Dx adapters only support Full Offload: Encrypted Overlay (where a Hypervisor controls IPsec offload - See for example OVS IPsec - https://docs.openvswitch.org/en/latest/tutorials/ipsec/) in a Linux OS with NVIDIA drivers.

This feature requires Linux kernel v6.6, or higher.

This feature is designed to enable IPsec full offload in switchdev mode. The `ip-xfrm` command is used to configure IPsec states and policies, and it is similar to legacy mode configuration. However, there are several limitations to the use of full offload in this mode:

1. Only IPsec Transport Mode and Tunnel Mode are supported.
2. The first IPsec TX state/policy is not allowed to be offloaded if any offloaded TC rule exists, and the same applies for the first RX state/policy. More specifically, IPsec RX/TX tables must be created before offloading any TC rule. For this reason, it is a common practice to configure IPsec rules before adding any TC rule.

Following is an example for IPsec configuration with a VXLan tunnel:

- Enable switchdev mode:

```
echo 1 > /sys/class/net/$PF0 /device/sriov_numvfs
echo 0000:08:00.2 > /sys/bus/pci/drivers/mlx5_core/unbind
devlink dev param set pci/0000:08:00.0 name flow_steering_mode value dmfs cmode runtime
devlink dev eswitch set pci/0000:08:00.0 mode switchdev
echo 0000:08:00.2 > /sys/bus/pci/drivers/mlx5_core/bind
```

- Configure PF/VF/REP netdevices, and place a VF in a namespace:

```
ifconfig $PF $LOCAL_TUN/16 up
ip l set dev $PF mtu 2000

ifconfig $REP up
ip netns add ns0
ip link set dev $VF netns ns0
ip netns exec ns0 ifconfig $VF $IP/16 up
```

- Configure IPsec states and policies:

```
ip xfrm state add src $LOCAL_TUN/16 dst $REMOTE_IP/16 proto esp spi 0xb29ed314 reqid 0xb29ed314 mode
transport aead 'rfc4106(gcm(aes))' 0x20f01f80a26f633d85617465686c32552c92c42f 128 offload packet dev $PF
dir out sel src $LOCAL_TUN/16 dst $REMOTE_IP/16 flag esn replay-window 64
ip xfrm state add src $REMOTE_IP/16 dst $LOCAL_TUN/16 proto esp spi 0xc35aa26e reqid 0xc35aa26e mode
transport aead 'rfc4106(gcm(aes))' 0x6cb228189b4c6e82e66e46920a2cde39187de4ba 128 offload packet dev $PF
dir in sel src $REMOTE_IP/16 dst $LOCAL_TUN/16 flag esn replay-window 64

ip xfrm policy add src $LOCAL_TUN dst $REMOTE_IP offload packet dev $PF dir out tmpl src $LOCAL_TUN/16 dst
$REMOTE_IP/16 proto esp reqid 0xb29ed314 mode transport priority 12
ip xfrm policy add src $REMOTE_IP dst $LOCAL_TUN offload packet dev $PF dir in tmpl src $REMOTE_IP/16 dst
$LOCAL_TUN/16 proto esp reqid 0xc35aa26e mode transport priority 12
```

- Configure Openvswitch:

```
ovs-vsctl add-br br-ovs
ovs-vsctl add-port br-ovs $REP
ovs-vsctl add-port br-ovs vxlan1 -- set interface vxlan1 type=vxlan options:local_ip=$LOCAL_TUN
options:remote_ip=$REMOTE_IP options:key=$VXLAN_ID options:dst_port=4789
```

3.3.1.21.1 IPsec Full Offload for RDMA Traffic

This IPsec Full Offload for RDMA Traffic option provides a significant performance improvement compared to the software IPsec counterpart, and enables the use of IPsec over RoCE packets, which are outside the network stack and cannot be used without full hardware offload. As a result, users can leverage the benefits of the IPsec protocol with RoCE V2, even when using SR-IOV VFs.

The configuration steps for this feature should be identical to the steps mentioned above, but if this feature is supported, the traffic that will be sent can also be RoCEV2 IPsec traffic.

To configure this feature:

1. Configure an SR-IOV VF normally, and add its OVS/TC rules.
2. Enable IPsec over VF. For more information, please see IPsec Functionality.
3. Configure IPsec policies and states on the relevant VF net device. This should be identical to the software configuration of IPsec rules, which can be done using one of the following implementation options:

Command	Offload Request Parameter
iproute2 ip xfrm	offload packet
libreswan	nic-offload=packet
strongswan	

For this feature to work, switchdev mode and dmfs steering mode must be enabled.

- The following is a full minimalistic configuration example using iproute, whereas PF0 is the netdevice PF, F0_REP is the VF representor, and NIC is the VF netdevice to configure IPsec over:

```
1.  echo 1 > /sys/class/net/$PF0 /device/sriov_numvfs
2.  echo 0000:08:00.2 > /sys/bus/pci/drivers/mlx5_core/unbind
3.  devlink dev eswitch set pci/0000:08:00.0 mode switchdev
4.  devlink dev param set pci/0000:08:00.0 name flow_steering_mode value dmfs cmode runtime
5.  devlink port function set pci/0000:08:00.0/1 ipsec_packet enable
6.  echo 0000:08:00.2 > /sys/bus/pci/drivers/mlx5_core/bind
7.  tc qdisc add dev $PF0 ingress
tc qdisc add dev $VF0_REP ingress
tc filter add dev $PF0 parent ffff: protocol 802.1q chain 0 flower vlan_id 10 vlan_ethtype 802.1q cvlan_id
5 action vlan pop action vlan pop  action mirred egress redirect dev $VF0_REP

tc filter add dev $VF0_REP parent ffff: protocol all chain 0 flower action vlan push protocol 802.1q id 5
 action vlan push protocol 802.1q id 10 action mirred egress redirect dev $PF0

8.  ifconfig $PF0 $PF_IP/24 up
ifconfig $NIC $LOC_IP/$SUB_NET up
ip link set dev $VF_REP up
9.  ip  xfrm state flush
ip  xfrm policy flush
```

- Configure ipsec states and policies:

```
#states
ip  -4 xfrm state add src $LOC_IP/$SUB_NET dst $REMOTE_IP/$SUB_NET proto esp spi 1000 reqid 10000 aead
'rfc4106(gcm(aes))' 0x010203047aeaca3f87d060a12f4a4487d5a5c335 128 mode transport sel src $LOC_IP dst
$REMOTE_IP offload packet dev $NIC dir out
ip  -4 xfrm state add src $REMOTE_IP/$SUB_NET dst $LOC_IP/$SUB_NET proto esp spi 1001 reqid 10001 aead
'rfc4106(gcm(aes))' 0x010203047aeaca3f87d060a12f4a4487d5a5c335 128 mode transport sel src $REMOTE_IP dst
$LOC_IP offload packet dev $NIC dir in
#policies
ip  -4 xfrm policy add src $LOC_IP dst $REMOTE_IP offload packet dev $NIC dir out tmpl src $LOC_IP/$SUB_NET
dst $REMOTE_IP/$SUB_NET proto esp reqid 10000 mode transport
ip  -4 xfrm policy add src $REMOTE_IP dst $LOC_IP offload packet dev $NIC dir in tmpl src $REMOTE_IP/
$SUB_NET dst $LOC_IP/$SUB_NET proto esp reqid 10001 mode transport
ip  -4 xfrm policy add src $REMOTE_IP dst $LOC_IP dir fwd tmpl src $REMOTE_IP/$SUB_NET dst $LOC_IP/$SUB_NET
proto esp reqid 10001 mode transport
```

Note that the configuration above is for one side only, yet IPsec must be configured for both sides in order for them to communicate properly. The configuration for the other side should be almost identical, but Step 9 would be configured in an asymmetrical way, meaning the first policy would look the following, and all other states/policies would be adjusted accordingly:

```
ip -4 xfrm state add src $LOC_IP/$SUB_NET dst $REMOTE_IP/$SUB_NET proto esp spi 1001 reqid 10001 aead
'rfc4106(gcm(aes))' 0x010203047aeaca3f87d060a12f4a4487d5a5c335 128 mode transport sel src $LOC_IP dst $REMOTE_IP
offload packet dev $NIC dir out
```

Once this step is completed, you can send any RoCE traffic of your choice between the two machines with configured IPsec. For example, `ibv_rc_pingpong -g 3 -d VF_device :` on one side, and `ibv_rc_pingpong -g 3 -d VF_device $IP_OF_OTHER_SIDE :` on the other side.

Finally, you can verify that the traffic was encrypted using IPsec by using the ipsec counters:

```
ethtool -S VF_NETDEV | grep ipsec
```

3.3.1.22 MACsec Full Offload

MACsec Full offload feature, also known as MACsec inline Full offload, enables the user to offload MACsec crypto encryption and decryption, MACsec headers encapsulation and decapsulation, and Anti replay operations to the hardware.

⚠ Hardware implementation supports GCM-AES & GCM-AES-XPN encryption schemes and is supported with ConnectX-7 onwards.

⚠ MACsec introduced in MOFED v5.9 requires a minimal Kernel version of 6.1.

To enable the feature, support in both kernel and adapter firmware is required.

For support in the kernel, make sure the following flags are set as follows:

- CONFIG_MACSEC=y
- CONFIG_MLX5_EN_MACSEC=y

For support in firmware use the following version:

- xx.34.0364 and up

3.3.1.22.1 Configurations

3.3.1.22.1.1 IProute2 Configuration

Configuring Physical Interface

Client side:

- ip address flush <physical_device>
- ip address add <client_physical_device_ip> dev <physical interface>
- ip link set dev <physical_device>up

Server side:

- ip address flush <physical_device>
- ip address add <server_physical_device_ip> dev <physical interface>
- ip link set dev <physical_device>up

Add MACsec Device

Client side:

- ip link add link <physical_device> <macsec_device> type macsec sci <client_sci> client on

Server side:

- ip link add link <physical_device> <macsec_device> type macsec sci <client_sci> client on

Offload MACsec Device

Client side:

- ip macsec offload <macsec_device> mac

Server side:

- ip macsec offload <macsec_device> mac

Add MACsec rules:

Client side:

- ip macsec add <macsec_device> tx sa <sa_num>pn <inital_packet_number>on key <client_key_id> <client_key>
- ip macsec add <macsec_device> rx sci <server_sci> on
- ip macsec add <macsec_device> rx sci <server_sci>sa <sa_num> pn <inital_packet_number> on key <server_key_id> <server_key>

Server side:

- ip macsec add <macsec_device> tx sa <sa_num>pn <inital_packet_number>on key <server_key_id> <server_key>
- ip macsec add <macsec_device> rx sci <client_sci> on

- ip macsec add <macsec_device> rx sci <client_sci>sa <sa_num> pn <inital_packet_number> on key <client_key_id> <client_key>

Configure MACsec Device IPs:

Client side:

- ip address flush <macsec_device>
- ip address add <client_macsec_device_ip> dev <macsec_device>
- ip link set dev <macsec_device> up

Server side:

- ip address flush <macsec_device>
- ip address add <server_macsec_device_ip> dev <macsec_device>
- ip link set dev <macsec_device> up

3.3.1.22.1.2 Configuration Example

Client side:

- ip address flush enp8s0f0
- ip address add 1.1.1.1/24 dev enp8s0f0
- ip link set dev enp8s0f0 up
- ip link add link enp8s0f0 macsec0 type macsec sci 1 encrypt on
- ip macsec offload macsec0 mac
- ip macsec add macsec0 tx sa 0 pn 1 on key 00 dffafc8d7b9a43d5b9a3dfbbf6a30c16
- ip macsec add macsec0 rx sci 2 on
- ip macsec add macsec0 rx sci 2 sa 0 pn 1 on key 00 ead3664f508eb06c40ac7104cdae4ce5
- ip address flush macsec0
- ip address add 2.2.2.1/24 dev macsec0
- ip link set dev macsec0 up

Server side:

- ip link del macsec0
- ip address flush enp8s0f0
- ip address add 1.1.1.2/24 dev enp8s0f0
- ip link set dev enp8s0f0 up
- ip link add link enp8s0f0 macsec0 type macsec sci 2 encrypt on
- ip macsec offload macsec0 mac
- ip macsec add macsec0 tx sa 0 pn 1 on key 00 ead3664f508eb06c40ac7104cdae4ce5
- ip macsec add macsec0 rx sci 1 on
- ip macsec add macsec0 rx sci 1 sa 0 pn 1 on key 00 dffafc8d7b9a43d5b9a3dfbbf6a30c16
- ip address flush macsec0
- ip address add 2.2.2.2/24 dev macsec0
- ip link set dev macsec0 up

- Use: "ip macsec show" command to check configuration

- To make sure traffic is offloaded, check MACsec counters: "`ethtool -S <physical_device> | grep macsec`"

Additional Resources

Linux Manual page: linux_manual

3.3.2 InfiniBand Network

The chapter contains the following sections:

3.3.2.1 InfiniBand Interface

3.3.2.1.1 Port Type Management

For information on port type management of ConnectX-4 and above adapter cards, please refer to Port Type Management/VPI Cards Configuration section.

3.3.2.1.2 RDMA Counters

- RDMA counters are available only through sysfs located under:
  - `# /sys/class/infiniband/<device>/ports/*/hw_counters/`
  - `# /sys/class/infiniband/<device>/hw_counters/`
  - `# /sys/class/infiniband/<device>/ports/*/counters`

For mlx5 port and RDMA counters, refer to the Understanding mlx5 Linux Counters Community post.

3.3.2.2 NVIDIA SM

NVIDIA SM is an InfiniBand compliant Subnet Manager (SM). It is provided as a fixed flow executable called "opensm", accompanied by a testing application called osmtest. NVIDIA SM implements an InfiniBand compliant SM according to the InfiniBand Architecture Specification chapters: Management Model, Subnet Management, and Subnet Administration.

3.3.2.2.1 OpenSM Application

OpenSM is an InfiniBand compliant Subnet Manager and Subnet Administrator that runs on top of the NVIDIA OFED stack. OpenSM performs the InfiniBand specification's required tasks for initializing InfiniBand hardware. One SM must be running for each InfiniBand subnet.
OpenSM defaults were designed to meet the common case usage on clusters with up to a few hundred nodes. Thus, in this default mode, OpenSM will scan the IB fabric, initialize it, and sweep occasionally for changes.
OpenSM attaches to a specific IB port on the local machine and configures only the fabric connected to it. (If the local machine has other IB ports, OpenSM will ignore the fabrics connected to those other ports). If no port is specified, opensm will select the first "best" available port. opensm can also present the available ports and prompt for a port number to attach to.
By default, the OpenSM run is logged to/var/log/opensm.log. All errors reported in this log file should be treated as indicators of IB fabric health issues. (Note that when a fatal and non-recoverable error occurs, OpenSM will exit). opensm.log should include the message "SUBNET UP" if OpenSM was able to set up the subnet correctly.

Syntax

```
opensm [OPTIONS]
```

For the complete list of OpenSM options, please run:

```
opensm --help / -h / -?
```

3.3.2.2.1.1 Environment Variables

The following environment variables control OpenSM behavior:

- OSM_TMP_DIR - controls the directory in which the temporary files generated by OpenSM are created. These files are: opensm-subnet.lst, opensm.fdbs, and opensm.mcfdbs. By default, this directory is /var/log.
- OSM_CACHE_DIR - opensm stores certain data to the disk such that subsequent runs are consistent. The default directory used is /var/cache/opensm. The following file is included in it:
  `guid2lid` – stores the LID range assigned to each GUID

3.3.2.2.1.2 Signaling

When OpenSM receives a HUP signal, it starts a new heavy sweep as if a trap has been received or a topology change has been found.
Also, SIGUSR1 can be used to trigger a reopen of /var/log/opensm.log for logrotate purposes.

3.3.2.2.1.3 Running OpenSM as Daemon

OpenSM can also run as daemon. To run OpenSM in this mode, enter:

```
host1# service opensmd start
```

3.3.2.2.2 osmtest

osmtest is a test program for validating the InfiniBand Subnet Manager and Subnet Administrator. osmtest provides a test suite for opensm. It can create an inventory file of all available nodes, ports, and PathRecords, including all their fields. It can also verify the existing inventory with all the object fields and matches it to a pre-saved one.
osmtest has the following test flows:

- Multicast Compliancy test
- Event Forwarding test
- Service Record registration test
- RMPP stress test
- Small SA Queries stress test

For further information, please refer to the tool's man page.

3.3.2.2.3 Partitions

OpenSM enables the configuration of partitions (PKeys) in an InfiniBand fabric. By default, OpenSM searches for the partitions configuration file under the name /etc/opensm/partitions.conf. To change this filename, you can use opensm with the '--Pconfig' or '-P' flags.
The default partition is created by OpenSM unconditionally, even when a partition configuration file does not exist or cannot be accessed.
The default partition has a P_Key value of 0x7fff. The port out of which runs OpenSM is assigned full membership in the default partition. All other end-ports are assigned partial membership.

3.3.2.2.3.1 File Format

⚠ Line content followed after '#' character is comment and ignored by parser.

General File Format

```
<Partition Definition>:\[<newline>\]<Partition Properties>
```

- <Partition Definition>:

```
[PartitionName][=PKey][,indx0][,ipoib_bc_flags][,defmember=full|limited]
```

where:

PartitionName	String, will be used with logging. When omitted empty string will be used.
PKey	P_Key value for this partition. Only low 15 bits will be used. When omitted will be auto-generated.
indx0	Indicates that this pkey should be inserted in block 0 index 0.
ipoib_bc_flags	Used to indicate/specify IPoIB capability of this partition.
defmember=full|limited|both	Specifies default membership for port GUID list. Default is limited.

ipoib_bc_flags are:

ipoib	Indicates that this partition may be used for IPoIB, as a result the IPoIB broadcast group will be created with the flags given, if any.
rate=<val>	Specifies rate for this IPoIB MC group (default is 3 (10GBps))
mtu=<val>	Specifies MTU for this IPoIB MC group (default is 4 (2048))
sl=<val>	Specifies SL for this IPoIB MC group (default is 0)
scope=<val>	Specifies scope for this IPoIB MC group (default is 2 (link local))

- <Partition Properties>:

```
\[<Port list>|<MCast Group>\]* | <Port list>
```

  - <Port List>:

```
<Port Specifier>[,<Port Specifier>]
```

    - <Port Specifier>:

```
<PortGUID>[=[full|limited|both]]
```

where

PortGUID	GUID of partition member EndPort. Hexadecimal numbers should start from 0x, decimal numbers are accepted too.

full, limited	Indicates full and/or limited membership for this both port. When omitted (or unrecognized) limited membership is assumed. Both indicate full and limited membership for this port.

- <MCast Group>:

```
mgid=gid[,mgroup_flag]*<newline>
```

where:

mgid=gid	gid specified is verified to be a Multicast address IP groups are verified to match the rate and mtu of the broadcast group. The P_Key bits of the mgid for IP groups are verified to either match the P_Key specified in by "Partition Definition" or if they are 0x0000 the P_Key will be copied into those bits.	
mgroup_flag	rate=<val>	Specifies rate for this MC group (default is 3 (10GBps))
	mtu=<val>	Specifies MTU for this MC group (default is 4 (2048))
	sl=<val>	Specifies SL for this MC group (default is 0)
	scope=<val>	Specifies scope for this MC group (default is 2 (link local)). Multiple scope settings are permitted for a partition. NOTE: This overwrites the scope nibble of the specified mgid. Furthermore specifying multiple scope settings will result in multiple MC groups being created.
	qkey=<val>	Specifies the Q_Key for this MC group (default: 0x0b1b for IP groups, 0 for other groups)
	tclass=<val>	Specifies tclass for this MC group (default is 0)
	FlowLabel=<val>	Specifies FlowLabel for this MC group (default is 0)

Note that values for rate, MTU, and scope should be specified as defined in the IBTA specification (for example, mtu=4 for 2048). To use 4K MTU, edit that entry to "mtu=5" (5 indicates 4K MTU to that specific partition).

PortGUIDs list:

```
PortGUID    GUID of partition member EndPort. Hexadecimal numbers should start from 0x, decimal numbers are
accepted too.
```

```
full or limited indicates full or limited membership for this port. When omitted (or unrecognized) limited
membership is assumed.
```

There are some useful keywords for PortGUID definition:

- 'ALL_CAS' means all Channel Adapter end ports in this subnet
- 'ALL_VCAS' means all virtual end ports in the subnet
- 'ALL_SWITCHES' means all Switch end ports in this subnet
- 'ALL_ROUTERS' means all Router end ports in this subnet
- 'SELF' means subnet manager's port. An empty list means that there are no ports in this partition

Notes:

- White space is permitted between delimiters ('=', ',',':',';').
- PartitionName does not need to be unique, PKey does need to be unique. If PKey is repeated then those partition configurations will be merged and the first PartitionName will be used (see the next note).
- It is possible to split partition configuration in more than one definition, but then PKey should be explicitly specified (otherwise different PKey values will be generated for those definitions).

Examples:

```
Default=0x7fff : ALL, SELF=full ;
Default=0x7fff : ALL, ALL_SWITCHES=full, SELF=full ;

NewPartition , ipoib : 0x123456=full, 0x3456789034=limi, 0x2134af2306 ;

YetAnotherOne = 0x300 : SELF=full ;
YetAnotherOne = 0x300 : ALL=limited ;

ShareIO = 0x80 , defmember=full : 0x123451, 0x123452;
# 0x123453, 0x123454 will be limited
ShareIO = 0x80 : 0x123453, 0x123454, 0x123455=full;
# 0x123456, 0x123457 will be limited
ShareIO = 0x80 : defmember=limited : 0x123456, 0x123457, 0x123458=full;
ShareIO = 0x80 , defmember=full : 0x123459, 0x12345a;
ShareIO = 0x80 , defmember=full : 0x12345b, 0x12345c=limited, 0x12345d;

# multicast groups added to default
Default=0x7fff,ipoib:
mgid=ff12:401b::0707,sl=1 # random IPv4 group
mgid=ff12:601b::16 # MLDv2-capable routers
mgid=ff12:401b::16 # IGMP
mgid=ff12:601b::2 # All routers
mgid=ff12::1,sl=1,Q_Key=0xDEADBEEF,rate=3,mtu=2 # random group
ALL=full;
```

The following rule is equivalent to how OpenSM used to run prior to the partition manager:

```
Default=0x7fff,ipoib:ALL=full;
```

3.3.2.2.4 Effect of Topology Changes

If a link is added or removed, OpenSM may not recalculate the routes that do not have to change. A route has to change if the port is no longer UP or no longer the MinHop. When routing changes are performed, the same algorithm for balancing the routes is invoked.
In the case of using the file-based routing, any topology changes are currently ignored. The 'file' routing engine just loads the LFTs from the file specified, with no reaction to real topology. Obviously, this will not be able to recheck LIDs (by GUID) for disconnected nodes, and LFTs for non-

existent switches will be skipped. Multicast is not affected by 'file' routing engine (this uses min hop tables).

3.3.2.2.5 Routing Algorithms

OpenSM offers the following routing engines:

1. Min Hop Algorithm
   Based on the minimum hops to each node where the path length is optimized.
2. UPDN Algorithm
   Based on the minimum hops to each node, but it is constrained to ranking rules. This algorithm should be chosen if the subnet is not a pure Fat Tree, and a deadlock may occur due to a loop in the subnet.
3. Fat-tree Routing Algorithm
   This algorithm optimizes routing for a congestion-free "shift" communication pattern. It should be chosen if a subnet is a symmetrical Fat Tree of various types, not just a K-ary-N-Tree: non-constant K, not fully staffed, and for any CBB ratio. Similar to UPDN, Fat Tree routing is constrained to ranking rules.
4. DOR Routing Algorithm
   Based on the Min Hop algorithm, but avoids port equalization except for redundant links between the same two switches. This provides deadlock free routes for hypercubes when the fabric is cabled as a hypercube and for meshes when cabled as a mesh.
5. Torus-2QoS Routing Algorithm
   Based on the DOR Unicast routing algorithm specialized for 2D/3D torus topologies. Torus-2QoS provides deadlock-free routing while supporting two quality of service (QoS) levels. Additionally, it can route around multiple failed fabric links or a single failed fabric switch without introducing deadlocks, and without changing path SL values granted before the failure.
6. Routing Chains
   Allows routing configuration of different parts of a single InfiniBand subnet by different routing engines. In the current release, minhop/updn/ftree/dor/torus-2QoS/pqft can be combined.

> ⚠ Please note that LASH Routing Algorithm is not supported.

MINHOP/UPDN/DOR routing algorithms are comprised of two stages:

1. MinHop matrix calculation. How many hops are required to get from each port to each LID. The algorithm to fill these tables is different if you run standard (min hop) or Up/Down. For standard routing, a "relaxation" algorithm is used to propagate min hop from every destination LID through neighbor switches. For Up/Down routing, a BFS from every target is used. The BFS tracks link direction (up or down) and avoid steps that will perform up after a down step was used.
2. Once MinHop matrices exist, each switch is visited and for each target LID a decision is made as to what port should be used to get to that LID. This step is common to standard and Up/Down routing. Each port has a counter counting the number of target LIDs going through it. When there are multiple alternative ports with same MinHop to a LID, the one with less

previously assigned ports is selected.
If LMC > 0, more checks are added. Within each group of LIDs assigned to same target port:

a. Use only ports which have same MinHop
b. First prefer the ones that go to different systemImageGuid (then the previous LID of the same LMC group)
c. If none, prefer those which go through another NodeGuid
d. Fall back to the number of paths method (if all go to same node).

3.3.2.2.5.1 Min Hop Algorithm

The Min Hop algorithm is invoked by default if no routing algorithm is specified. It can also be invoked by specifying '-R minhop'.
The Min Hop algorithm is divided into two stages: computation of min-hop tables on every switch and LFT output port assignment. Link subscription is also equalized with the ability to override based on port GUID. The latter is supplied by:

```
-i <equalize-ignore-guids-file>
-ignore-guids <equalize-ignore-guids-file>
```

This option provides the means to define a set of ports (by GUIDs) that will be ignored by the link load equalization algorithm.
LMC awareness routes based on a (remote) system or on a switch basis.

3.3.2.2.5.2 UPDN Algorithm

The UPDN algorithm is designed to prevent deadlocks from occurring in loops of the subnet. A loop-deadlock is a situation in which it is no longer possible to send data between any two hosts connected through the loop. As such, the UPDN routing algorithm should be sent if the subnet is not a pure Fat Tree, and one of its loops may experience a deadlock (due, for example, to high pressure).
The UPDN algorithm is based on the following main stages:

1. Auto-detect root nodes - based on the CA hop length from any switch in the subnet, a statistical histogram is built for each switch (hop num vs the number of occurrences). If the histogram reflects a specific column (higher than others) for a certain node, then it is marked as a root node. Since the algorithm is statistical, it may not find any root nodes. The list of the root nodes found by this auto-detect stage is used by the ranking process stage.

   ⚠ The user can override the node list manually.

   ⚠ If this stage cannot find any root nodes, and the user did not specify a GUID list file, OpenSM defaults back to the Min Hop routing algorithm.

2. Ranking process - All root switch nodes (found in stage 1) are assigned a rank of 0. Using the BFS algorithm, the rest of the switch nodes in the subnet are ranked incrementally. This ranking aids in the process of enforcing rules that ensure loop-free paths.
3. Min Hop Table setting - after ranking is done, a BFS algorithm is run from each (CA or switch) node in the subnet. During the BFS process, the FDB table of each switch node traversed by

BFS is updated, in reference to the starting node, based on the ranking rules and GUID values.

At the end of the process, the updated FDB tables ensure loop-free paths through the subnet.

UPDN Algorithm Usage

Activation through OpenSM:

- Use '-R updn' option (instead of old '-u') to activate the UPDN algorithm.
- Use '-a <root_guid_file>' for adding an UPDN GUID file that contains the root nodes for ranking. If the `-a' option is not used, OpenSM uses its auto-detect root nodes algorithm.

Notes on the GUID list file:

- A valid GUID file specifies one GUID in each line. Lines with an invalid format will be discarded
- The user should specify the root switch GUIDs

3.3.2.2.5.3 Fat-tree Routing Algorithm

The fat-tree algorithm optimizes routing for "shift" communication pattern. It should be chosen if a subnet is a symmetrical or almost symmetrical fat-tree of various types. It supports not just K- ary-N-Trees, by handling for non-constant K, cases where not all leafs (CAs) are present, any Constant Bisectional Ratio (CBB)ratio. As in UPDN, fat-tree also prevents credit-loop-dead- locks.
If the root GUID file is not provided ('a' or '-root_guid_file' options), the topology has to be pure fat-tree that complies with the following rules:

- Tree rank should be between two and eight (inclusively)
- Switches of the same rank should have the same number of UP-going port groups, unless they are root switches, in which case the shouldn't have UP-going ports at all.
  Note: Ports that are connected to the same remote switch are referenced as ‘port group’.
- Switches of the same rank should have the same number of DOWN-going port groups, unless they are leaf switches.
- Switches of the same rank should have the same number of ports in each UP-going port group.
- Switches of the same rank should have the same number of ports in each DOWN-going port group.
- All the CAs have to be at the same tree level (rank).

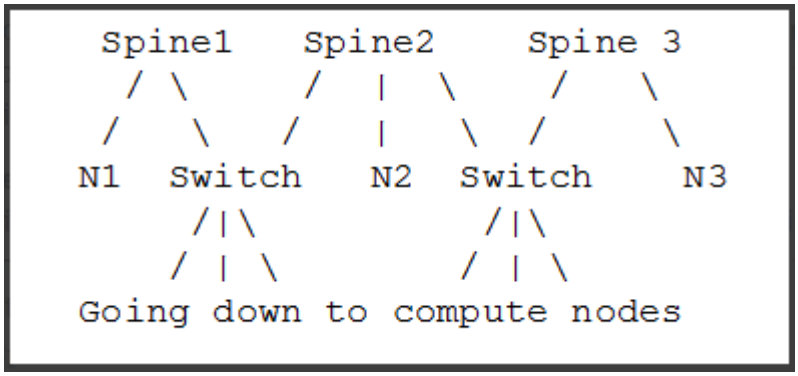
If the root GUID file is provided, the topology does not have to be pure fat-tree, and it should only comply with the following rules:

- Tree rank should be between two and eight (inclusively)
- All the Compute Nodes have to be at the same tree level (rank). Note that non-compute node CAs are allowed here to be at different tree ranks.
  Note: List of compute nodes (CNs) can be specified using ‘-u’ or ‘--cn_guid_file’ OpenSM options.

Topologies that do not comply cause a fallback to min-hop routing. Note that this can also occur on link failures which cause the topology to no longer be a "pure" fat-tree.
Note that although fat-tree algorithm supports trees with non-integer CBB ratio, the routing will not be as balanced as in case of integer CBB ratio. In addition to this, although the algorithm allows leaf

switches to have any number of CAs, the closer the tree is to be fully populated, the more effective the "shift" communication pattern will be. In general, even if the root list is provided, the closer the topology to a pure and symmetrical fat-tree, the more optimal the routing will be.
The algorithm also dumps the compute node ordering file (opensm-ftree-ca-order.dump) in the same directory where the OpenSM log resides. This ordering file provides the CN order that may be used to create efficient communication pattern, that will match the routing tables.

Routing between non-CN Nodes
The use of the io_guid_file option allows non-CN nodes to be located on different levels in the fat tree. In such case, it is not guaranteed that the Fat Tree algorithm will route between two non-CN nodes. In the scheme below, N1, N2 , and N3 are non-CN nodes. Although all the CN have routes to and from them, there will not necessarily be a route between N1,N2 and N3. Such routes would require to use at least one of the switches the wrong way around.



To solve this problem, a list of non-CN nodes can be specified by \'-G\' or \'--io_guid_file\' option. These nodes will be allowed to use switches the wrong way around a specific number of times (specified by \'-H\' or \'--max_reverse_hops\'. With the proper max_reverse_hops and io_guid_file values, you can ensure full connectivity in the Fat Tree. In the scheme above, with a max_reverse_hop of 1, routes will be instantiated between N1<->N2 and N2<->N3. With a max_reverse_hops value of 2, N1,N2 and N3 will all have routes between them.

> ⚠ Using max_reverse_hops creates routes that use the switch in a counter-stream way. This option should never be used to connect nodes with high bandwidth traffic between them! It should only be used to allow connectivity for HA purposes or similar. Also having routes the other way around can cause credit loops.

Activation through OpenSM

Use '-R ftree' option to activate the fat-tree algorithm.

> ⚠ LMC > 0 is not supported by fat-tree routing. If this is specified, the default routing algorithm is invoked instead.

3.3.2.2.5.4 DOR Routing Algorithm

The Dimension Order Routing algorithm is based on the Min Hop algorithm and so uses shortest paths. Instead of spreading traffic out across different paths with the same shortest distance, it

chooses among the available shortest paths based on an ordering of dimensions. Each port must be consistently cabled to represent a hypercube dimension or a mesh dimension. Paths are grown from a destination back to a source using the lowest dimension (port) of available paths at each step. This provides the ordering necessary to avoid deadlock. When there are multiple links between any two switches, they still represent only one dimension and traffic is balanced across them unless port equalization is turned off. In the case of hypercubes, the same port must be used throughout the fabric to represent the hypercube dimension and match on both ends of the cable. In the case of meshes, the dimension should consistently use the same pair of ports, one port on one end of the cable, and the other port on the other end, continuing along the mesh dimension.
Use '-R dor' option to activate the DOR algorithm.

3.3.2.2.5.5 Torus-2QoS Routing Algorithm

Torus-2QoS is a routing algorithm designed for large-scale 2D/3D torus fabrics. The torus-2QoS routing engine can provide the following functionality on a 2D/3D torus:

- Free of credit loops routing
- Two levels of QoS, assuming switches support 8 data VLs
- Ability to route around a single failed switch, and/or multiple failed links, without:
    - introducing credit loops
    - changing path SL values
- Very short run times, with good scaling properties as fabric size increases

Unicast Routing

Torus-2 QoS is a DOR-based algorithm that avoids deadlocks that would otherwise occur in a torus using the concept of a dateline for each torus dimension. It encodes into a path SL which datelines the path crosses as follows:

```
sl = 0;
for (d = 0; d < torus_dimensions; d++)
/* path_crosses_dateline(d) returns 0 or 1 */
sl |= path_crosses_dateline(d) << d;
```

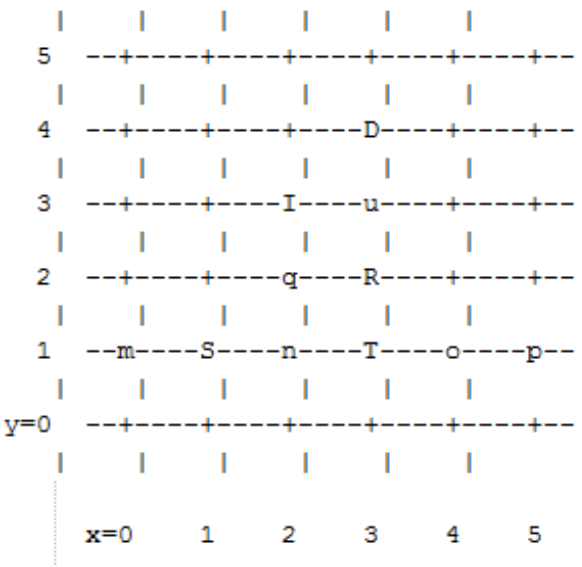
For a 3D torus, that leaves one SL bit free, which torus-2 QoS uses to implement two QoS levels. Torus-2 QoS also makes use of the output port dependence of switch SL2VL maps to encode into one VL bit the information encoded in three SL bits. It computes in which torus coordinate direc- tion each inter-switch link "points", and writes SL2VL maps for such ports as follows:

```
for (sl = 0; sl < 16; sl ++)
/* cdir(port) reports which torus coordinate direction a switch port
* "points" in, and returns 0, 1, or 2 */
sl2vl(iport,oport,sl) = 0x1 & (sl >> cdir(oport));
```

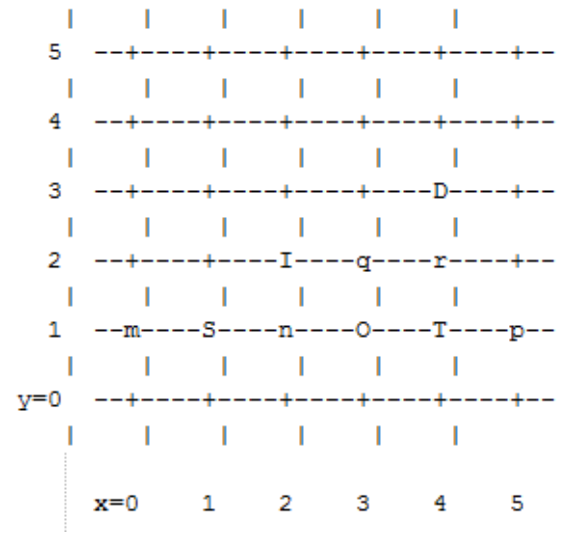
Thus, on a pristine 3D torus, i.e., in the absence of failed fabric switches, torus-2 QoS consumes 8 SL values (SL bits 0-2) and 2 VL values (VL bit 0) per QoS level to provide deadlock-free routing on a 3D torus. Torus-2 QoS routes around link failure by "taking the long way around" any 1D ring interrupted by a link failure. For example, consider the 2D 6x5 torus below, where switches are denoted by [+a-zA-Z]:

For a pristine fabric the path from S to D would be S-n-T-r-D. In the event that either link S-n or n-T has failed, torus-2QoS would use the path S-m-p-o-T-r-D.
Note that it can do this without changing the path SL value; once the 1D ring m-S-n-T-o-p-m has been broken by failure, path segments using it cannot contribute to deadlock, and the x-direction dateline (between, say, x=5 and x=0) can be ignored for path segments on that ring. One result of this is that torus-2QoS can route around many simultaneous link failures, as long as no 1D ring is broken into disjoint segments. For example, if links n-T and T-o have both failed, that ring has been broken into two disjoint segments, T and o-p-m-S-n. Torus-2QoS checks for such issues, reports if they are found, and refuses to route such fabrics.
Note that in the case where there are multiple parallel links between a pair of switches, torus-2QoS will allocate routes across such links in a round-robin fashion, based on ports at the path destination switch that are active and not used for inter-switch links. Should a link that is one of several such parallel links fail, routes are redistributed across the remaining links. When the last of such a set of parallel links fails, traffic is rerouted as described above.
Handling a failed switch under DOR requires introducing into a path at least one turn that would be otherwise "illegal", i.e. not allowed by DOR rules. Torus-2QoS will introduce such a turn as close as possible to the failed switch in order to route around it. n the above example, suppose switch T has failed, and consider the path from S to D. Torus-2QoS will produce the path S-n-I-r-D, rather than the S-n-T-r-D path for a pristine torus, by introducing an early turn at n. Normal DOR rules will cause traffic arriving at switch I to be forwarded to switch r; for traffic arriving from I due to the "early" turn at n, this will generate an "illegal" turn at I.
Torus-2QoS will also use the input port dependence of SL2VL maps to set VL bit 1 (which would be otherwise unused) for y-x, z-x, and z-y turns, i.e., those turns that are illegal under DOR. This causes the first hop after any such turn to use a separate set of VL values, and prevents deadlock in the presence of a single failed switch. For any given path, only the hops after a turn that is illegal under DOR can contribute to a credit loop that leads to deadlock. So in the example above with failed switch T, the location of the illegal turn at I in the path from S to D requires that any credit loop caused by that turn must encircle the failed switch at T. Thus the second and later hops after the illegal turn at I (i.e., hop r-D) cannot contribute to a credit loop because they cannot be used to construct a loop encircling T. The hop I-r uses a separate VL, so it cannot contribute to a credit loop encircling T. Extending this argument shows that in addition to being capable of routing around a single switch failure without introducing deadlock, torus-2QoS can also route around multiple failed

switches on the condition they are adjacent in the last dimension routed by DOR. For example, consider the following case on a 6x6 2D torus:



Suppose switches T and R have failed, and consider the path from S to D. Torus-2QoS will generate the path S-n-q-I-u-D, with an illegal turn at switch I, and with hop I-u using a VL with bit 1 set. As a further example, consider a case that torus-2QoS cannot route without deadlock: two failed switches adjacent in a dimension that is not the last dimension routed by DOR; here the failed switches are O and T:

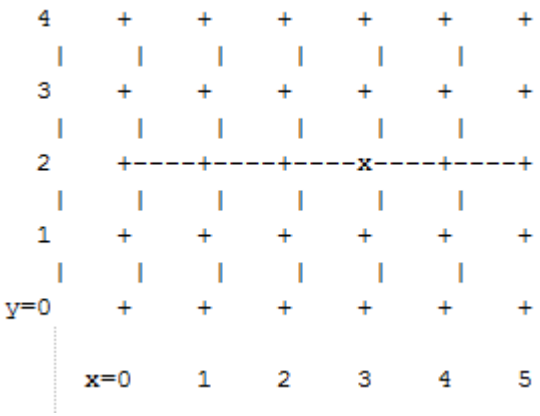


In a pristine fabric, torus-2QoS would generate the path from S to D as S-n-O-T-r-D. With failed switches O and T, torus-2QoS will generate the path S-n-I-q-r-D, with an illegal turn at switch I, and

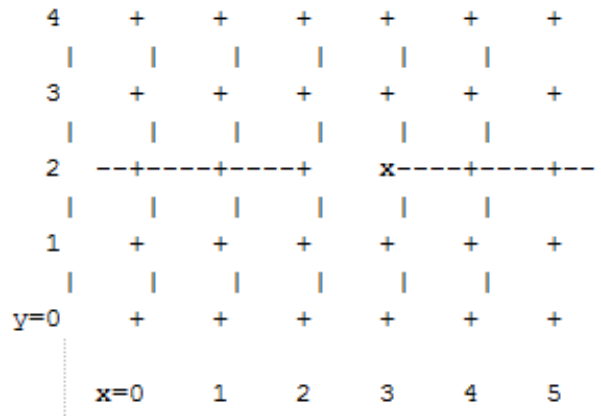
with hop I-q using a VL with bit 1 set. In contrast to the earlier examples, the second hop after the illegal turn, q-r, can be used to construct a credit loop encircling the failed switches.

Multicast Routing

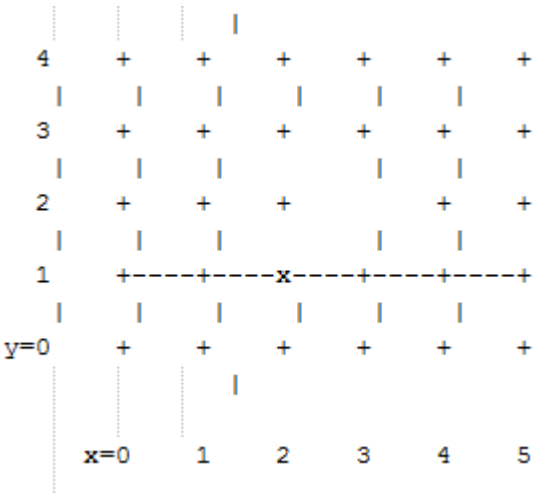
Since torus-2QoS uses all four available SL bits, and the three data VL bits that are typically available in current switches, there is no way to use SL/VL values to separate multicast traffic from unicast traffic. Thus, torus-2QoS must generate multicast routing such that credit loops cannot arise from a combination of multicast and unicast path segments. It turns out that it is possible to construct spanning trees for multicast routing that have that property. For the 2D 6x5 torus example above, here is the full-fabric spanning tree that torus-2QoS will construct, where "x" is the root switch and each "+" is a non-root switch:



For multicast traffic routed from root to tip, every turn in the above spanning tree is a legal DOR turn. For traffic routed from tip to root, and some traffic routed through the root, turns are not legal DOR turns. However, to construct a credit loop, the union of multicast routing on this spanning tree with DOR unicast routing can only provide 3 of the 4 turns needed for the loop. In addition, if none of the above spanning tree branches crosses a dateline used for unicast credit loop avoidance on a torus, and if multicast traffic is confined to SL 0 or SL 8 (recall that torus-2QoS uses SL bit 3 to differentiate QoS level), then multicast traffic also cannot contribute to the "ring" credit loops that are otherwise possible in a torus. Torus-2QoS uses these ideas to create a master spanning tree. Every multicast group spanning tree will be constructed as a subset of the master tree, with the same root as the master tree. Such multicast group spanning trees will in general not be optimal for groups which are a subset of the full fabric. However, this compromise must be made to enable support for two QoS levels on a torus while preventing credit loops. In the presence of link or switch failures that result in a fabric for which torus-2QoS can generate credit-loop-free unicast routes, it is also possible to generate a master spanning tree for multicast that retains the required properties. For example, consider that same 2D 6x5 torus, with the link from (2,2) to (3,2) failed. Torus-2QoS will generate the following master spanning tree:



Two things are notable about this master spanning tree. First, assuming the x dateline was between x=5 and x=0, this spanning tree has a branch that crosses the dateline. However, just as for unicast, crossing a dateline on a 1D ring (here, the ring for y=2) that is broken by a failure cannot contribute to a torus credit loop. Second, this spanning tree is no longer optimal even for multicast groups that encompass the entire fabric. That, unfortunately, is a compromise that must be made to retain the other desirable properties of torus-2QoS routing. In the event that a single switch fails, torus-2QoS will generate a master spanning tree that has no "extra" turns by appropriately selecting a root switch. In the 2D 6x5 torus example, assume now that the switch at (3,2) (i.e., the root for a pristine fabric), fails. Torus-2QoS will generate the following master spanning tree for that case:



Assuming the dateline was between y=4 and y=0, this spanning tree has a branch that crosses a dateline. However, this cannot contribute to credit loops as it occurs on a 1D ring (the ring for x=3) that is broken by failure, as in the above example.

Torus Topology Discovery

The algorithm used by torus-2QoS to construct the torus topology from the undirected graph representing the fabric requires that the radix of each dimension be configured via torus-2QoS.conf. It also requires that the torus topology be "seeded"; for a 3D torus this requires configuring four

switches that define the three coordinate directions of the torus. Given this starting information, the algorithm is to examine the cube formed by the eight switch locations bounded by the corners (x,y,z) and (x+1,y+1,z+1). Based on switches already placed into the torus topology at some of these locations, the algorithm examines 4-loops of inter-switch links to find the one that is consistent with a face of the cube of switch locations and adds its switches to the discovered topology in the correct locations.
Because the algorithm is based on examining the topology of 4-loops of links, a torus with one or more radix-4 dimensions requires extra initial seed configuration. See torus-2QoS.conf(5) for details. Torus-2QoS will detect and report when it has an insufficient configuration for a torus with radix-4 dimensions.
In the event the torus is significantly degraded, i.e., there are many missing switches or links, it may happen that torus-2QoS is unable to place into the torus some switches and/or links that were discovered in the fabric, and will generate a warning in that case. A similar condition occurs if torus-2QoS is misconfigured, i.e., the radix of a torus dimension as configured does not match the radix of that torus dimension as wired, and many switches/links in the fabric will not be placed into the torus.

Quality Of Service Configuration

OpenSM will not program switches and channel adapters with SL2VL maps or VL arbitration configuration unless it is invoked with -Q. Since torus-2QoS depends on such functionality for correct operation, always invoke OpenSM with -Q when torus-2QoS is in the list of routing engines. Any quality of service configuration method supported by OpenSM will work with torus-2QoS, subject to the following limitations and considerations. For all routing engines supported by OpenSM except torus-2QoS, there is a one-to-one correspondence between QoS level and SL. Torus-2QoS can only support two quality of service levels, so only the high-order bit of any SL value used for unicast QoS configuration will be honored by torus-2QoS. For multicast QoS configuration, only SL values 0 and 8 should be used with torus-2QoS.
Since SL to VL map configuration must be under the complete control of torus-2QoS, any configuration via qos_sl2vl, qos_swe_sl2vl, etc., must and will be ignored, and a warning will be generated. Torus-2QoS uses VL values 0-3 to implement one of its supported QoS levels, and VL values 4-7 to implement the other. Hard-to-diagnose application issues may arise if traffic is not delivered fairly across each of these two VL ranges. Torus-2QoS will detect and warn if VL arbitration is configured unfairly across VLs in the range 0-3, and also in the range 4-7. Note that the default OpenSM VL arbitration configuration does not meet this constraint, so all torus-2QoS users should configure VL arbitration via qos_vlarb_high, qos_vlarb_low, etc.

Operational Considerations

Any routing algorithm for a torus IB fabric must employ path SL values to avoid credit loops. As a result, all applications run over such fabrics must perform a path record query to obtain the correct path SL for connection setup. Applications that use rdma_cm for connection setup will automatically meet this requirement.
If a change in fabric topology causes changes in path SL values required to route without credit loops, in general, all applications would need to repath to avoid message deadlock. Since torus-2QoS has the ability to reroute after a single switch failure without changing path SL values, repathing by running applications is not required when the fabric is routed with torus-2QoS. Torus-2QoS can provide unchanging path SL values in the presence of subnet manager failover provided that all OpenSM instances have the same idea of dateline location. See torus- 2QoS.conf(5) for details. Torus-2QoS will detect configurations of failed switches and links that prevent routing that is free of credit loops and will log warnings and refuse to route. If "no_fall- back" was configured in the list of OpenSM routing engines, then no other routing engine will attempt to route

the fabric. In that case, all paths that do not transit the failed components will continue to work, and the subset of paths that are still operational will continue to remain free of credit loops. OpenSM will continue to attempt to route the fabric after every sweep interval and after any change (such as a link up) in the fabric topology. When the fabric components are repaired, full functionality will be restored. In the event OpenSM was configured to allow some other engine to route the fabric if torus-2QoS fails, then credit loops and message deadlock are likely if torus-2QoS had previously routed the fabric successfully. Even if the other engine is capable of routing a torus without credit loops, applications that built connections with path SL values granted under torus-2QoS will likely experience message deadlock under routing generated by a different engine, unless they repath. To verify that a torus fabric is routed free of credit loops, use `ibdmchk` to analyze data collected via ibdiagnet `-` vlr.

Torus-2QoS Configuration File Syntax

The file torus-2QoS.conf contains configuration information that is specific to the OpenSM routing engine torus-2QoS. Blank lines and lines where the first non-whitespace character is "#" are ignored. A token is any contiguous group of non-whitespace characters. Any tokens on a line following the recognized configuration tokens described below are ignored.

```
[torus|mesh] x_radix[m|M|t|T] y_radix[m|M|t|T] z_radix[m|M|t|T]
```

Either torus or mesh must be the first keyword in the configuration and sets the topology that torus-2QoS will try to construct. A 2D topology can be configured by specifying one of x_radix, y_radix, or z_radix as 1. An individual dimension can be configured as mesh (open) or torus (looped) by suffixing its radix specification with one of m, M, t, or T. Thus, "mesh 3T 4 5" and "torus 3 4M 5M" both specify the same topology.
Note that although torus-2QoS can route mesh fabrics, its ability to route around failed components is severely compromised on such fabrics. A failed fabric components very likely to cause a disjoint ring; see UNICAST ROUTING in torus-2QoS(8).

```
xp_link sw0_GUID sw1_GUID
yp_link sw0_GUID sw1_GUID
zp_link sw0_GUID sw1_GUID
xm_link sw0_GUID sw1_GUID
ym_link sw0_GUID sw1_GUID
zm_link sw0_GUID sw1_GUID
```

These keywords are used to seed the torus/mesh topology. For example, "xp_link 0x2000 0x2001" specifies that a link from the switch with node GUID 0x2000 to the switch with node GUID 0x2001 would point in the positive x direction, while "xm_link 0x2000 0x2001" specifies that a link from the switch with node GUID 0x2000 to the switch with node GUID 0x2001 would point in the negative x direction. All the link keywords for a given seed must specify the same "from" switch.
In general, it is not necessary to configure both the positive and negative directions for a given coordinate; either is sufficient. However, the algorithm used for topology discovery needs extra information for torus dimensions of radix four (see TOPOLOGY DISCOVERY in torus-2QoS(8)). For such cases, both the positive and negative coordinate directions must be specified.
Based on the topology specified via the torus/mesh keyword, torus-2QoS will detect and log when it has insufficient seed configuration.

```
GUIDx_dateline position
y_dateline position
z_dateline position
```

In order for torus-2QoS to provide the guarantee that path SL values do not change under any conditions for which it can still route the fabric, its idea of dateline position must not change relative to physical switch locations. The dateline keywords provide the means to configure such behavior.

The dateline for a torus dimension is always between the switch with coordinate 0 and the switch with coordinate radix-1 for that dimension. By default, the common switch in a torus seed is taken as the origin of the coordinate system used to describe switch location. The position parameter for a dateline keyword moves the origin (and hence the dateline) the specified amount relative to the common switch in a torus seed.

```
next_seed
```

If any of the switches used to specify a seed were to fail torus-2QoS would be unable to complete topology discovery successfully. The next_seed keyword specifies that the following link and dateline keywords apply to a new seed specification.
For maximum resiliency, no seed specification should share a switch with any other seed specification. Multiple seed specifications should use dateline configuration to ensure that torus-2QoS can grant path SL values that are constant, regardless of which seed was used to initiate topology discovery.
portgroup_max_ports max_ports - This keyword specifies the maximum number of parallel inter-switch links, and also the maximum number of host ports per switch, that torus-2QoS can accommodate. The default value is 16. Torus-2QoS will log an error message during topology discovery if this parameter needs to be increased. If this keyword appears multiple times, the last instance prevails.
port_order p1 p2 p3 ... - This keyword specifies the order in which CA ports on a destination switch are visited when computing routes. When the fabric contains switches connected with multiple parallel links, routes are distributed in a round-robin fashion across such links, and so changing the order that CA ports are visited changes the distribution of routes across such links. This may be advantageous for some specific traffic patterns.
The default is to visit CA ports in increasing port order on destination switches. Duplicate values in the list will be ignored.

Example:

```
# Look for a 2D (since x radix is one) 4x5 torus.
torus 1 4 5
# y is radix-4 torus dimension, need both
# ym_link and yp_link configuration.
yp_link 0x200000 0x200005 # sw @ y=0,z=0 -> sw @ y=1,z=0
ym_link 0x200000 0x20000f # sw @ y=0,z=0 -> sw @ y=3,z=0
# z is not radix-4 torus dimension, only need one of
# zm_link or zp_link configuration.
zp_link 0x200000 0x200001 # sw @ y=0,z=0 -> sw @ y=0,z=1
next_seed
yp_link 0x20000b 0x200010 # sw @ y=2,z=1 -> sw @ y=3,z=1
ym_link 0x20000b 0x200006 # sw @ y=2,z=1 -> sw @ y=1,z=1
zp_link 0x20000b 0x20000c # sw @ y=2,z=1 -> sw @ y=2,z=2
y_dateline -2 # Move the dateline for this seed
z_dateline -1 # back to its original position.
# If OpenSM failover is configured, for maximum resiliency
# one instance should run on a host attached to a switch
# from the first seed, and another instance should run
# on a host attached to a switch from the second seed.
# Both instances should use this torus-2QoS.conf to ensure
# path SL values do not change in the event of SM failover.
# port_order defines the order on which the ports would be
# chosen for routing.
port_order 7 10 8 11 9 12 25 28 26 29 27 30
```

3.3.2.2.5.6 Routing Chains

The routing chains feature is offering a solution that enables one to configure different parts of the fabric and define a different routing engine to route each of them. The routings are done in a sequence (hence the name "chains") and any node in the fabric that is configured in more than one part is left with the routing updated by the last routing engine it was a part of.

Configuring Routing Chains

To configure routing chains:

1. Define the port groups.
2. Define topologies based on previously defined port groups.
3. Define configuration files for each routing engine.
4. Define routing engine chains over previously defined topologies and configuration files.

Defining Port Groups

The basic idea behind the port groups is the ability to divide the fabric into sub-groups and give each group an identifier that can be used to relate to all nodes in this group. The port groups is a separate feature from the routing chains but is a mandatory prerequisite for it. In addition, it is used to define the participants in each of the routing algorithms.

Defining a Port Group Policy File

In order to define a port group policy file, set the parameter 'pgrp_policy_file' in the OpenSM configuration file.
pgrp_policy_file /etc/opensm/conf/port_groups_policy_file

Configuring a Port Group Policy

The port groups policy file details the port groups in the fabric. The policy file should be composed of one or more paragraphs that define a group. Each paragraph should begin with the line 'port-group' and end with the line 'end-port-group'.
For example:

```
port-group
…port group qualifiers…
end-port-group
```

Port Group Qualifiers

Unlike the port group's beginning and end which do not require a colon, all qualifiers must end with a colon (':'). Also - a colon is a predefined mark that must not be used inside qualifier values. The inclusion of a colon in the name or the use of a port group will result in the policy's failure.

Rule Qualifier

Parameter	Description	Example
`name`	Each group must have a name. Without a name qualifier, the policy fails.	`name: grp1`
`use`	'use' is an optional qualifier that one can define in order to describe the usage of this port group (if undefined, an empty string is used as a default).	`use: first port group`

There are several qualifiers used to describe a rule that determines which ports will be added to the group. Each port group may include one or more rules out of the rules described in the below table (at least one rule must be defined for each port group).

Parameter	Description	Example
`guid list`	Comma separated list of GUIDs to include in the group. If no specific physical ports were configured, all physical ports of the guid are chosen. However, for each guid, one can detail specific physical ports to be included in the group. This can be done using the following syntax: • Specify a specific port in a guid to be chosen port-guid: 0x283@3 • Specify a specific list of ports in a guid to be chosen port-guid: 0x286@1/5/7 • Specify a specific range of ports in a guid to be chosen port-guid: 0x289@2-5 • Specify a list of specific ports and ports ranges in a guid to be chosen port-guid: 0x289@2-5/7/9-13/18 • Complex rule port-guid: 0x283@5-8/12/14, 0x286, 0x289/6/ 8/12	`port-guid: 0x283, 0x286, 0x289`
`port guid range`	It is possible to configure a range of guids to be chosen to the group. However, while using the range qualifier, it is impossible to detail specific physical ports. Note: A list of ranges cannot be specified. The below example is invalid and will cause the policy to fail: port-guid-range: 0x283-0x289, 0x290- 0x295	`port-guid-range: 0x283-0x289`
`port name`	One can configure a list of hostnames as a rule. Hosts with a node description that is built out of these hostnames will be chosen. Since the node description contains the network card index as well, one might also specify a network card index and a physical port to be chosen. For example, the given configuration will cause only physical port 2 of a host with the node description 'kuku HCA-1' to be chosen. port and hca_idx parameters are optional. If the port is unspecified, all physical ports are chosen. If hca_idx is unspecified, all card numbers are chosen. Specifying a hostname is mandatory. One can configure a list of hostname/ port/hca_idx sets in the same qualifier as follows: port-name: hostname=kuku; port=2; hca_idx=1 , hostname=host1; port=3, hostname=host2 Note: port-name qualifier is not relevant for switches, but for HCA's only.	`port-name: host-name=kuku; port=2; hca_idx=1`
`port regexp`	One can define a regular expression so that only nodes with a matching node description will be chosen to the group. Note: This example shows how to choose nodes which their node description starts with 'SW'.	`port-regexp: SW`
	It is possible to specify one physical port to be chosen for matching nodes (there is no option to define a list or a range of ports). The given example will cause only nodes that match physical port 3 to be added to the group.	`port-regexp: SW:3`

Parameter	Description	Example
`union rule`	It is possible to define a rule that unites two different port groups. This means that all ports from both groups will be included in the united group.	`union-rule: grp1, grp2`
`subtract rule`	One can define a rule that subtracts one port group from another. The given rule, for example, will cause all the ports which are a part of grp1, but not included in grp2, to be chosen. In subtraction (unlike union), the order does matter, since the purpose is to subtract the second group from the first one. There is no option to define more than two groups for union/subtraction. However, one can unite/subtract groups which are a union or a subtraction themselves, as shown in the port groups policy file example.	`subtract-rule: grp1, grp2`

Predefined Port Groups

There are 3 predefined, automatically created port groups that are available for use, yet cannot be defined in the policy file (if a group in the policy is configured with the name of one of these predefined groups, the policy fails) -

- ALL - a group that includes all nodes in the fabric
- ALL_SWITCHES - a group that includes all switches in the fabric
- ALL_CAS - a group that includes all HCAs in the fabric
- ALL_ROUTERS - a group that includes all routers in the fabric (supported in OpenSM starting from v4.9.0)

Port Groups Policy Examples

```
port-group
name: grp3
use: Subtract of groups grp1 and grp2
subtract-rule: grp1, grp2
end-port-group

port-group
name: grp1
port-guid: 0x281, 0x282, 0x283
end-port-group

port-group
name: grp2
port-guid-range: 0x282-0x286
port-name: hostname=server1 port=1
end-port-group

port-group
name: grp4
port-name: hostname=kika port=1 hca_idx=1
end-port-group

port-group
name: grp3
union-rule: grp3, grp4
end-port-group
```

Defining a Topologies Policy File

In order to define a topology policy file, set the parameter 'topo_policy_file' in the OpenSM configuration file.

```
topo_policy_file /etc/opensm/conf/topo_policy_file.cfg
```

Configuring a Topology Policy

The topologies policy file details a list of topologies. The policy file should be composed of one or more paragraphs which define a topology. Each paragraph should begin with the line 'topol- ogy' and end with the line 'end-topology'.
For example:

```
topology
…topology qualifiers…
end-topology
```

Topology Qualifiers

> Unlike topology and end-topology which do not require a colon, all qualifiers must end with a colon (':'). Also - a colon is a predefined mark that must not be used inside qualifier values. An inclusion of a column in the qualifier values will result in the policy's failure.

All topology qualifiers are mandatory. Absence of any of the below qualifiers will cause the policy parsing to fail.

Topology Qualifiers

Parameter	Description	Example
`id`	Topology ID. Legal Values - any positive value. Must be unique.	`id: 1`
`sw-grp`	Name of the port group that includes all switches and switch ports to be used in this topology.	`sw-grp: ys_switches`
`hca-grp`	Name of the port group that includes all HCA's to be used in this topology.	`hca-grp: ys_hosts`

Configuration File per Routing Engine

Each engine in the routing chain can be provided by its own configuration file. Routing engine configuration file is the fraction of parameters defined in the main OpenSM configuration file. Some rules should be applied when defining a particular configuration file for a routing engine:

- Parameters that are not specified in specific routing engine configuration file are inherited from the main OpenSM configuration file.
- The following configuration parameters are taking effect only in the main OpenSM configuration file:
  - qos and qos_* settings like (vl_arb, sl2vl, etc.)
  - lmc
  - routing_engine

Defining a Routing Chain Policy File

In order to define a port group policy file, set the parameter 'rch_policy_file' in the OpenSM configuration file.

```
rch_policy_file /etc/opensm/conf/chains_policy_file
```

First Routing Engine in the Chain

The first unicast engine in a routing chain must include all switches and HCAs in the fabric (topology id must be 0). The path-bit parameter value is path-bit 0 and it cannot be changed.

Configuring a Routing Chains Policy

The routing chains policy file details the routing engines (and their fallback engines) used for the fabric's routing. The policy file should be composed of one or more paragraphs which defines an engine (or a fallback engine). Each paragraph should begin with the line 'unicast-step' and end with the line 'end-unicast-step'.
For example:

```
unicast-step
…routing engine qualifiers…
end-unicast-step
```

Routing Engine Qualifiers

> ⚠ Unlike unicast-step and end-unicast-step which do not require a colon, all qualifiers must end with a colon (':'). Also - a colon is a predefined mark that must not be used inside qualifier values. An inclusion of a colon in the qualifier values will result in the policy's failure.

Parameter	Description	Example
`id`	'id' is mandatory. Without an ID qualifier for each engine, the policy fails. • Legal values - size_t value (0 is illegal). • The engines in the policy chain are set according to an ascending id order, so it is highly crucial to verify that the id that is given to the engines match the order in which you would like the engines to be set.	`is: 1`
`engine`	This is a mandatory qualifier that describes the routing algorithm used within this unicast step. Currently, on the first phase of routing chains, legal values are minhop/ftree/ updn.	`engine: minhop`
`use`	This is an optional qualifier that enables one to describe the usage of this unicast step. If undefined, an empty string is used as a default.	`use: ftree routing for for yellow stone nodes`
`config`	This is an optional qualifier that enables one to define a separate OpenSM config file for a specific unicast step. If undefined, all parameters are taken from main OpenSM configuration file.	`config: / etc/config/ opensm2.cfg`
`topology`	Define the topology that this engine uses. • Legal value - id of an existing topology that is defined in topologies policy (or zero that represents the entire fabric and not a specific topology). • Default value - If unspecified, a routing engine will relate to the entire fabric (as if topology zero was defined). • Notice: The first routing engine (the engine with the lowest id) MUST be configured with topology: 0 (entire fabric) or else, the routing chain parser will fail.	`topology: 1`

Parameter	Description	Example
`fallback-to`	This is an optional qualifier that enables one to define the current unicast step as a fallback to another unicast step. This can be done by defining the id of the unicast step that this step is a fallback to. • If undefined, the current unicast step is not a fallback. • If the value of this qualifier is a non-existent engine id, this step will be ignored. • A fallback step is meaningless if the step it is a fallback to did not fail. • It is impossible to define a fallback to a fall- back step (such definition will be ignored)	`-`
`path-bit`	This is an optional qualifier that enables one to define a specific lid offset to be used by the current unicast step. Setting lmc > 0 in main OpenSM configuration file is a prerequisite for assigning specific path-bit for the routing engine. Default value is 0 (if path-bit is not specified)	`Path-bit: 1`

Dump Files per Routing Engine

Each routing engine on the chain will dump its own data files if the appropriate log_flags is set (for instance 0x43).

The files that are dumped by each engine are:

- opensm-lid-matrix.dump
- opensm-lfts.dump
- opensm.fdbs
- opensm-subnet.lst

These files should contain the relevant data for each engine topology.

⚠ sl2vl and mcfdbs files are dumped only once for the entire fabric and NOT by every routing engine.

- Each engine concatenates its ID and routing algorithm name in its dump files names, as follows:
  - opensm-lid-matrix.2.minhop.dump
  - opensm.fdbs.3.ftree
  - opensm-subnet.4.updn.lst
- In case that a fallback routing engine is used, both the routing engine that failed and the fallback engine that replaces it, dump their data.
  If, for example, engine 2 runs ftree and it has a fallback engine with 3 as its id that runs minhop, one should expect to find 2 sets of dump files, one for each engine:
  - opensm-lid-matrix.2.ftree.dump
  - opensm-lid-matrix.3.minhop.dump
  - opensm.fdbs.2.ftree
  - opensm.fdbs.3.munhop

3.3.2.2.6 Unicast Routing Cache

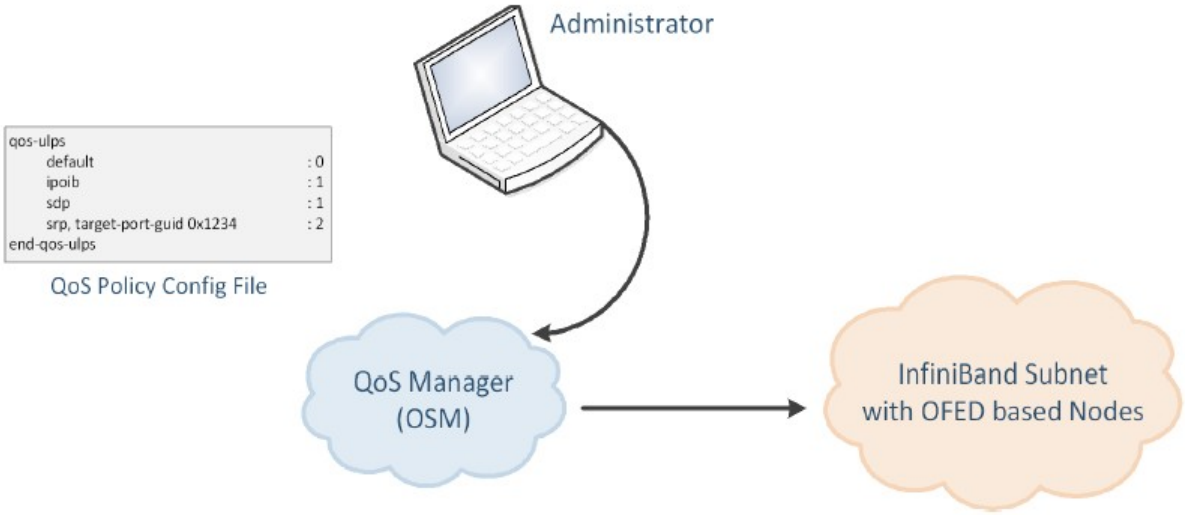
Unicast routing cache prevents routing recalculation (which is a heavy task in a large cluster) when no topology change was detected during the heavy sweep, or when the topology change does not

require new routing calculation (for example, when one or more CAs/RTRs/leaf switches going down, or one or more of these nodes coming back after being down).

3.3.2.2.7 Quality of Service Management in OpenSM

When Quality of Service (QoS) in OpenSM is enabled (using the '-Q' or '--qos' flags), OpenSM looks for a QoS Policy file. During fabric initialization and at every heavy sweep, OpenSM parses the QoS policy file, applies its settings to the discovered fabric elements, and enforces the provided policy on client requests. The overall flow for such requests is as follows:

- The request is matched against the defined matching rules such that the QoS Level definition is found
- Given the QoS Level, a path(s) search is performed with the given restrictions imposed by that level



There are two ways to define QoS policy:

- Advanced – the advanced policy file syntax provides the administrator various ways to match a PathRecord/MultiPathRecord (PR/MPR) request, and to enforce various QoS constraints on the requested PR/MPR
- Simple – the simple policy file syntax enables the administrator to match PR/MPR requests by various ULPs and applications running on top of these ULPs

3.3.2.2.7.1 Advanced QoS Policy File

The QoS policy file has the following sections:

1. Port Groups (denoted by port-groups) - this section defines zero or more port groups that can be referred later by matching rules (see below). Port group lists ports by:
   - Port GUID
   - Port name, which is a combination of NodeDescription and IB port number

- PKey, which means that all the ports in the subnet that belong to partition with a given PKey belong to this port group
- Partition name, which means that all the ports in the subnet that belong to partition with a given name belong to this port group
- Node type, where possible node types are: CA, SWITCH, ROUTER, ALL, and SELF (SM's port).

2. QoS Setup (denoted by qos-setup) - this section describes how to set up SL2VL and VL Arbitration tables on various nodes in the fabric. However, this is not supported in OFED. SL2VL and VLArb tables should be configured in the OpenSM options file (default location - /var/cache/opensm/opensm.opts).

3. QoS Levels (denoted by qos-levels) - each QoS Level defines Service Level (SL) and a few optional fields:
   - MTU limit
   - Rate limit
   - PKey
   - Packet lifetime

   When path(s) search is performed, it is done with regards to restriction that these QoS Level parameters impose. One QoS level that is mandatory to define is a DEFAULT QoS level. It is applied to a PR/MPR query that does not match any existing match rule. Similar to any other QoS Level, it can also be explicitly referred by any match rule.

- QoS Matching Rules (denoted by qos-match-rules) - each PathRecord/MultiPathRecord query that OpenSM receives is matched against the set of matching rules. Rules are scanned in order of appearance in the QoS policy file such as the first match takes precedence.
  Each rule has a name of QoS level that will be applied to the matching query. A default QoS level is applied to a query that did not match any rule.
  Queries can be matched by:
  - Source port group (whether a source port is a member of a specified group)
  - Destination port group (same as above, only for destination port)
  - PKey
  - QoS class
  - Service ID

  To match a certain matching rule, PR/MPR query has to match ALL the rule's criteria. However, not all the fields of the PR/MPR query have to appear in the matching rule.
  For instance, if the rule has a single criterion - Service ID, it will match any query that has this Service ID, disregarding rest of the query fields. However, if a certain query has only Service ID (which means that this is the only bit in the PR/MPR component mask that is on), it will not match any rule that has other matching criteria besides Service ID.

3.3.2.2.7.2 Simple QoS Policy Definition

Simple QoS policy definition comprises of a single section denoted by qos-ulps. Similar to the advanced QoS policy, it has a list of match rules and their QoS Level, but in this case a match rule has only one criterion - its goal is to match a certain ULP (or a certain application on top of this ULP) PR/MPR request, and QoS Level has only one constraint - Service Level (SL).

The simple policy section may appear in the policy file in combine with the advanced policy, or as a stand-alone policy definition. See more details and list of match rule criteria below.

3.3.2.2.7.3 Policy File Syntax Guidelines

- Leading and trailing blanks, as well as empty lines, are ignored, so the indentation in the example is just for better readability.
- Comments are started with the pound sign (#) and terminated by EOL.
- Any keyword should be the first non-blank in the line, unless it's a comment.
- Keywords that denote section/subsection start have matching closing keywords.
- Having a QoS Level named "DEFAULT" is a must - it is applied to PR/MPR requests that did not match any of the matching rules.
- Any section/subsection of the policy file is optional.

3.3.2.2.7.4 Examples of Advanced Policy Files

As mentioned earlier, any section of the policy file is optional, and the only mandatory part of the policy file is a default QoS Level.
Here is an example of the shortest policy file:

```
qos-levels
    qos-level
        name: DEFAULT
        sl: 0
    end-qos-level
end-qos-levels
```

Port groups section is missing because there are no match rules, which means that port groups are not referred anywhere, and there is no need defining them. And since this policy file doesn't have any matching rules, PR/MPR query will not match any rule, and OpenSM will enforce default QoS level. Essentially, the above example is equivalent to not having a QoS policy file at all.
The following example shows all the possible options and keywords in the policy file and their syntax:

```
#
# See the comments in the following example.
# They explain different keywords and their meaning.
#
port-groups

    port-group # using port GUIDs
        name: Storage
        # "use" is just a description that is used for logging
        #  Other than that, it is just a comment
        use: SRP Targets
        port-guid: 0x10000000000001, 0x10000000000005-0x1000000000FFFA
        port-guid: 0x1000000000FFFF
    end-port-group

    port-group
        name: Virtual Servers
        # The syntax of the port name is as follows:
        #   "node_description/Pnum".
        # node_description is compared to the NodeDescription of the node,
        # and "Pnum" is a port number on that node.
        port-name: “vs1 HCA-1/P1, vs2 HCA-1/P1”
    end-port-group

    # using partitions defined in the partition policy
    port-group
        name: Partitions
        partition: Part1
        pkey: 0x1234
    end-port-group

    # using node types: CA, ROUTER, SWITCH, SELF (for node that runs SM)
    # or ALL (for all the nodes in the subnet)
    port-group
        name: CAs and SM
        node-type: CA, SELF
    end-port-group

end-port-groups
```

```
qos-setup
    # This section of the policy file describes how to set up SL2VL and VL
    # Arbitration tables on various nodes in the fabric.
    # However, this is not supported in OFED - the section is parsed
    # and ignored. SL2VL and VLArb tables should be configured in the
    # OpenSM options file (by default - /var/cache/opensm/opensm.opts).
end-qos-setup

qos-levels

    # Having a QoS Level named "DEFAULT" is a must - it is applied to
    # PR/MPR requests that didn't match any of the matching rules.
    qos-level
        name: DEFAULT
        use: default QoS Level
        sl: 0
    end-qos-level

    # the whole set: SL, MTU-Limit, Rate-Limit, PKey, Packet Lifetime
    qos-level
        name: WholeSet
        sl: 1
        mtu-limit: 4
        rate-limit: 5
        pkey: 0x1234
        packet-life: 8
    end-qos-level

end-qos-levels

# Match rules are scanned in order of their appearance in the policy file.
# First matched rule takes precedence.
qos-match-rules

    # matching by single criteria: QoS class
    qos-match-rule
        use: by QoS class
        qos-class: 7-9,11
        # Name of qos-level to apply to the matching PR/MPR
        qos-level-name: WholeSet
    end-qos-match-rule

    # show matching by destination group and service id
    qos-match-rule
        use: Storage targets
        destination: Storage
        service-id: 0x10000000000001, 0x10000000000008-0x1000000000FFF
        qos-level-name: WholeSet
    end-qos-match-rule

    qos-match-rule
        source: Storage
        use: match by source group only
        qos-level-name: DEFAULT
    end-qos-match-rule
    qos-match-rule
        use: match by all parameters
        qos-class: 7-9,11
        source: Virtual Servers
        destination: Storage
        service-id: 0x0000000000010000-0x000000000001FFFF
        pkey: 0x0F00-0x0FFF
        qos-level-name: WholeSet
    end-qos-match-rule
end-qos-match-rules
```

3.3.2.2.7.5 Simple QoS Policy - Details and Examples

Simple QoS policy match rules are tailored for matching ULPs (or some application on top of a ULP) PR/MPR requests. This section has a list of per-ULP (or per-application) match rules and the SL that should be enforced on the matched PR/MPR query.
Match rules include:

- Default match rule that is applied to PR/MPR query that didn't match any of the other match rules
- IPoIB with a default PKey
- IPoIB with a specific PKey
- Any ULP/application with a specific Service ID in the PR/MPR query
- Any ULP/application with a specific PKey in the PR/MPR query
- Any ULP/application with a specific target IB port GUID in the PR/MPR query

Since any section of the policy file is optional, as long as basic rules of the file are kept (such as no referring to nonexistent port group, having default QoS Level, etc), the simple policy section (qos-

ulps) can serve as a complete QoS policy file.
The shortest policy file in this case would be as follows:

```
qos-ulps
    default  : 0 #default SL
end-qos-ulps
```

It is equivalent to the previous example of the shortest policy file, and it is also equivalent to not having policy file at all. Below is an example of simple QoS policy with all the possible keywords:

```
qos-ulps
default                 :0 # default SL
sdp, port-num 30000     :0 # SL for application running on
                           # top of SDP when a destination
                           # TCP/IPport is 30000
sdp, port-num 10000-20000     : 0
sdp                     :1 # default SL for any other
                           # application running on top of SDP
rds                     :2 # SL for RDS traffic
ipoib, pkey 0x0001      :0 # SL for IPoIB on partition with
                           # pkey 0x0001
ipoib                   :4 # default IPoIB partition,
                           # pkey=0x7FFF
any, service-id 0x6234:6 # match any PR/MPR query with a
                           # specific Service ID
any, pkey 0x0ABC        :6 # match any PR/MPR query with a
                           # specific PKey
srp, target-port-guid 0x1234   : 5 # SRP when SRP Target is located
                           # on a specified IB port GUID
any, target-port-guid 0x0ABC-0xFFFFF : 6 # match any PR/MPR query
                           # with a specific target port GUID
end-qos-ulps
```

Similar to the advanced policy definition, matching of PR/MPR queries is done in order of appearance in the QoS policy file such as the first match takes precedence, except for the "default" rule, which is applied only if the query didn't match any other rule. All other sections of the QoS policy file take precedence over the qos-ulps section. That is, if a policy file has both qos-match-rules and qos-ulps sections, then any query is matched first against the rules in the qos-match-rules section, and only if there was no match, the query is matched against the rules in qos-ulps section. Note that some of these match rules may overlap, so in order to use the simple QoS definition effectively, it is important to understand how each of the ULPs is matched.

3.3.2.2.7.6 IPoIB

IPoIB query is matched by PKey or by destination GID, in which case this is the GID of the multicast group that OpenSM creates for each IPoIB partition.
Default PKey for IPoIB partition is 0x7fff, so the following three match rules are equivalent:

```
ipoib:<SL>ipoib, pkey 0x7fff : <SL>
any, pkey 0x7fff : <SL>
```

3.3.2.2.7.7 SRP

Service ID for SRP varies from storage vendor to vendor, thus SRP query is matched by the target IB port GUID. The following two match rules are equivalent:

```
srp, target-port-guid 0x1234 : <SL>
any, target-port-guid 0x1234 : <SL>
```

Note that any of the above ULPs might contain target port GUID in the PR query, so in order for these queries not to be recognized by the QoS manager as SRP, the SRP match rule (or any match

rule that refers to the target port GUID only) should be placed at the end of the qos-ulps match rules.

3.3.2.2.7.8 MPI

SL for MPI is manually configured by an MPI admin. OpenSM is not forcing any SL on the MPI traffic, which explains why it is the only ULP that did not appear in the qos-ulps section.

3.3.2.2.7.9 SL2VL Mapping and VL Arbitration

OpenSM cached options file has a set of QoS related configuration parameters, that are used to configure SL2VL mapping and VL arbitration on IB ports. These parameters are:

- Max VLs: the maximum number of VLs that will be on the subnet
- High limit: the limit of High Priority component of VL Arbitration table (IBA 7.6.9)
- VLArb low table: Low priority VL Arbitration table (IBA 7.6.9) template
- VLArb high table: High priority VL Arbitration table (IBA 7.6.9) template
- SL2VL: SL2VL Mapping table (IBA 7.6.6) template. It is a list of VLs corresponding to SLs 0-15 (Note that VL15 used here means drop this SL).

There are separate QoS configuration parameters sets for various target types: CAs, routers, switch external ports, and switch's enhanced port 0. The names of such parameters are prefixed by "qos_<type>_" string. Here is a full list of the currently supported sets:

- qos_ca_ —QoS configuration parameters set for CAs.
- qos_rtr_ —parameters set for routers.
- qos_sw0_ —parameters set for switches' port 0.
- qos_swe_ —parameters set for switches' external ports.

Here's the example of typical default values for CAs and switches' external ports (hard-coded in OpenSM initialization):

```
qos_ca_max_vls 15
qos_ca_high_limit 0
qos_ca_vlarb_high 0:4,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0,13:0,14:0
qos_ca_vlarb_low 0:0,1:4,2:4,3:4,4:4,5:4,6:4,7:4,8:4,9:4,10:4,11:4,12:4,13:4,14:4
qos_ca_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
qos_swe_max_vls 15
qos_swe_high_limit 0
qos_swe_vlarb_high 0:4,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0,13:0,14:0
qos_swe_vlarb_low 0:0,1:4,2:4,3:4,4:4,5:4,6:4,7:4,8:4,9:4,10:4,11:4,12:4,13:4,14:4
qos_swe_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
```

VL arbitration tables (both high and low) are lists of VL/Weight pairs. Each list entry contains a VL number (values from 0-14), and a weighting value (values 0-255), indicating the number of 64 byte units (credits) which may be transmitted from that VL when its turn in the arbitration occurs. A weight of 0 indicates that this entry should be skipped. If a list entry is programmed for VL15 or for a VL that is not supported or is not currently configured by the port, the port may either skip that entry or send from any supported VL for that entry.
Note, that the same VLs may be listed multiple times in the High or Low priority arbitration tables, and, further, it can be listed in both tables. The limit of high-priority VLArb table (qos_<type>_high_limit) indicates the number of high-priority packets that can be transmitted without an opportunity to send a low-priority packet. Specifically, the number of bytes that can be sent is high_limit times 4K bytes.
A high_limit value of 255 indicates that the byte limit is unbounded.

⚠ If the 255 value is used, the low priority VLs may be starved.

A value of 0 indicates that only a single packet from the high-priority table may be sent before an opportunity is given to the low-priority table.
Keep in mind that ports usually transmit packets of size equal to MTU. For instance, for 4KB MTU a single packet will require 64 credits, so in order to achieve effective VL arbitration for packets of 4KB MTU, the weighting values for each VL should be multiples of 64.
Below is an example of SL2VL and VL Arbitration configuration on subnet:

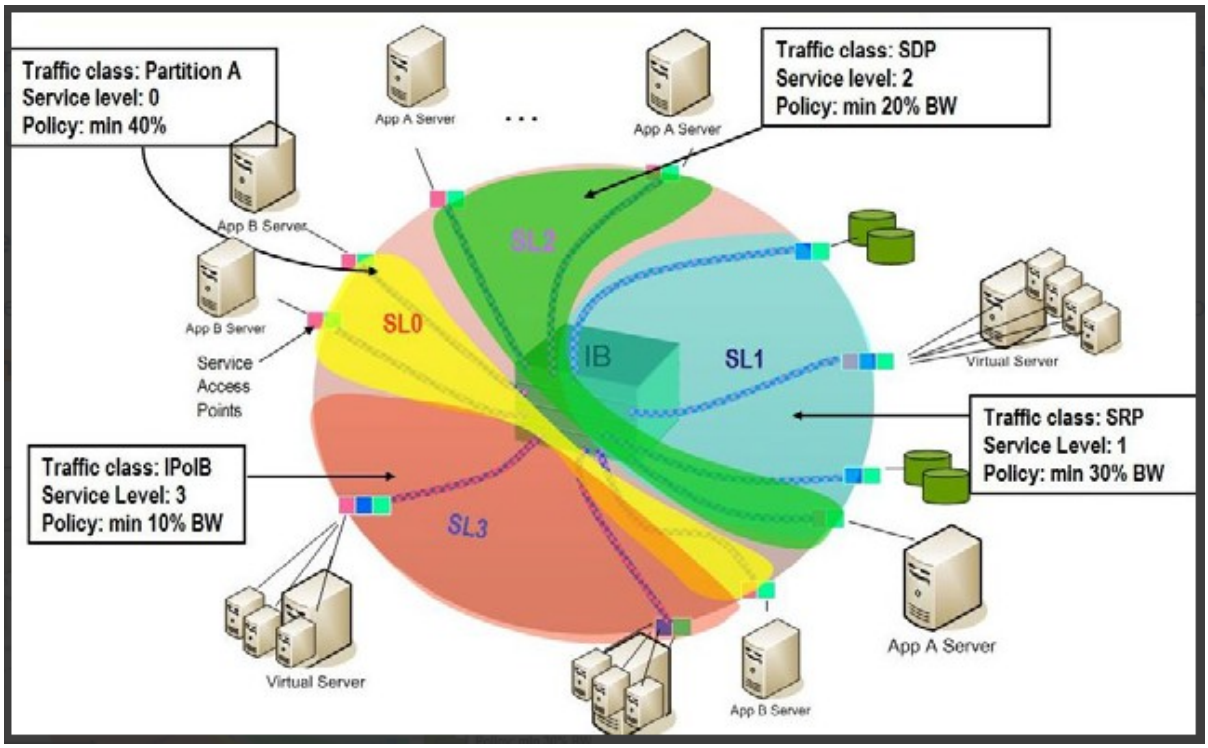
```
qos_ca_max_vls 15
qos_ca_high_limit 6
qos_ca_vlarb_high 0:4
qos_ca_vlarb_low 0:0,1:64,2:128,3:192,4:0,5:64,6:64,7:64
qos_ca_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
qos_swe_max_vls 15
qos_swe_high_limit 6
qos_swe_vlarb_high 0:4
qos_swe_vlarb_low 0:0,1:64,2:128,3:192,4:0,5:64,6:64,7:64
qos_swe_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
```

In this example, there are 8 VLs configured on subnet: VL0 to VL7. VL0 is defined as a high priority VL, and it is limited to 6 x 4KB = 24KB in a single transmission burst. Such configuration would suilt VL that needs low latency and uses small MTU when transmitting packets. Rest of VLs are defined as low priority VLs with different weights, while VL4 is effectively turned off.

3.3.2.2.7.10 Deployment Example

The figure below shows an example of an InfiniBand subnet that has been configured by a QoS manager to provide different service levels for various ULPs.

QoS Deployment on InfiniBand Subnet Example



QoS Configuration Examples

The following are examples of QoS configuration for different cluster deployments. Each example provides the QoS level assignment and their administration via OpenSM configuration files.

Typical HPC Example: MPI and Lustre

Assignment of QoS Levels

- MPI
    - Separate from I/O load
    - Min BW of 70%
- Storage Control (Lustre MDS)
    - Low latency
- Storage Data (Lustre OST)
    - Min BW 30%

Administration

- MPI is assigned an SL via the command line
  `host1# mpirun -sl 0`
- OpenSM QoS policy file

```
qos-ulps
   default                                    :0 # default SL (for MPI)
   any, target-port-guid OST1,OST2,OST3,OST4  :1 # SL for Lustre OST
   any, target-port-guid MDS1,MDS2            :2 # SL for Lustre MDS
```

```
end-qos-ulps
```

Note: In this policy file example, replace OST* and MDS* with the real port GUIDs.

- OpenSM options file

```
qos_max_vls 8
qos_high_limit 0
qos_vlarb_high 2:1
qos_vlarb_low 0:96,1:224
qos_sl2vl 0,1,2,3,4,5,6,7,15,15,15,15,15,15,15,15
```

EDC SOA (2-tier): IPoIB and SRP

The following is an example of QoS configuration for a typical enterprise data center (EDC) with service oriented architecture (SOA), with IPoIB carrying all application traffic and SRP used for storage.

QoS Levels

- Application traffic
  - IPoIB (UD and CM) and SDP
  - Isolated from storage
  - Min BW of 50%
- SRP
  - Min BW 50%
  - Bottleneck at storage nodes

Administration

- OpenSM QoS policy file

```
qos-ulps
   default                                 :0
   ipoib                                   :1
   sdp                                     :1
   srp, target-port-guid SRPT1,SRPT2,SRPT3     :2
end-qos-ulps
```

Note: In this policy file example, replace SRPT* with the real SRP Target port GUIDs.

- OpenSM options file

```
qos_max_vls 8
qos_high_limit 0
qos_vlarb_high 1:32,2:32
qos_vlarb_low 0:1,
qos_sl2vl 0,1,2,3,4,5,6,7,15,15,15,15,15,15,15,15
```

EDC (3-tier): IPoIB, RDS, SRP

The following is an example of QoS configuration for an enterprise data center (EDC), with IPoIB carrying all application traffic, RDS for database traffic, and SRP used for storage.

QoS Levels

- Management traffic (ssh)
  - IPoIB management VLAN (partition A)
  - Min BW 10%
- Application traffic

        - IPoIB application VLAN (partition B)
        - Isolated from storage and database
        - Min BW of 30%
    - Database Cluster traffic
        - RDS
        - Min BW of 30%
    - SRP
        - Min BW 30%
        - Bottleneck at storage nodes

Administration

- OpenSM QoS policy file

```
qos-ulps
   default                                  :0
   ipoib, pkey 0x8001                       :1
   ipoib, pkey 0x8002                       :2
   rds                                      :3
   srp, target-port-guid SRPT1, SRPT2, SRPT3   :4
end-qos-ulps
```

  Note: In the following policy file example, replace SRPT* with the real SRP Initiator port GUIDs.

- OpenSM options file

```
qos_max_vls 8
qos_high_limit 0
qos_vlarb_high 1:32,2:96,3:96,4:96
qos_vlarb_low 0:1
qos_sl2vl 0,1,2,3,4,5,6,7,15,15,15,15,15,15,15,15
```

- Partition configuration file

```
Default=0x7fff,ipoib : ALL=full;PartA=0x8001, sl=1, ipoib : ALL=full;
```

3.3.2.2.7.11 Enhanced QoS

Enhanced QoS provides a higher resolution of QoS at the service level (SL). Users can configure rate limit values per SL for physical ports, virtual ports, and port groups, using enhanced_qos_policy_file configuration parameter.
Valid values of this parameter:

- Full path to the policy file through which Enhanced QoS Manager is configured
- "null" - to disable the Enhanced QoS Manager (default value)

To enable Enhanced QoS Manager, QoS must be enabled in OpenSM.

Enhanced QoS Policy File

The policy file is comprised of three sections:

- BW_NAMES: Used to define bandwidth setting and name (currently, rate limit is the only setting). Bandwidth names can be used in BW_RULES and VPORT_BW_RULES sections.
  Bandwidth names are defined using the syntax:
  `<name> = <rate limit in 1Mbps units>`
  Example: `My_bandwidth = 50`
- BW_RULES: Used to define the rules that map the bandwidth setting to a specific SL of a specific GUID.
  Bandwidth rules are defined using the syntax:
  `<guid>|<port group name> = <sl id>:<bandwidth name>, <sl id>:<bandwidth name>…`
  Examples:
  `0x2c90000000025 = 5:My_bandwidth, 7:My_bandwidth`
  `Port_grp1 = 3:My_bandwidth, 9:My_bandwidth`
- VPORT_BW_RULES: Used to define the rules that map the bandwidth setting to a specific SL of a specific virtual port GUID.
  Bandwidth rules are defined using the syntax:
  `<guid>= <sl id>:<bandwidth name>, <sl id>:<bandwidth name>…`
  Examples:
  `0x2c90000000026= 5:My_bandwidth, 7:My_bandwidth`

Special Keywords

- Keyword “all” allows setting a rate limit of all SLs to some BW for a specific physical or virtual port. It is possible to combine “all” with specific SL rate limits.
  Example:
  `0x2c90000000025 = all:BW1,SL3:BW2`
  In this case, SL3 will be assigned BW2 rate limit, while the rest of SLs get BW1 rate limit.
- "default" is a well-known name which can be used to define a default rule used for any GUID with no defined rule.
  If no default rule is defined, any GUID without a specific rule will be configured with unlimited rate limit for all SLs.
  Keyword “all” is also applicable to the default rule. Default rule is local to each section.

Special Subnet Manager Configuration Options

New SM configuration option enhanced_qos_vport0_unlimit_default_rl was added to opensm.conf.

The possible values for this configuration option are:

- TRUE: For specific virtual port0 GUID, SLs not mentioned in bandwidth rule will be set to unlimited bandwidth (0) regardless of the default rule of the VPORT_BW_RULES section. Virtual port0 GUIDs not mentioned in VPORT_BW_SECTION will be set to unlimited BW on all SLs.
- FALSE: The GUID of virtual port0 is treated as any other virtual port in VPORT_BW_SECTION.

SM should be signaled by HUP once the option is changed.

Default: TRUE

Notes

- When rate limit is set to 0, it means that the bandwidth is unlimited.
- Any unspecified SL in a rule will be set to 0 (unlimited) rate limit automatically if no default rule is specified.
- Failure to complete policy file parsing leads to an undefined behavior. User must confirm no relevant error messages in SM log in order to ensure Enhanced QoS Manager is configured properly.
- A file with only 'BW_NAMES' and 'BW_RULES' keywords configures the network with an unlimited rate limit.
- HCA physical port GUID can be specified in BW_RULES and VPORT_BW_RULES sections.
- In BW_RULES section, the rate limit assigned to a specific SL will limit the total BW that can be sent through the PF on a given SL.
- In VPORT_BW_RULES section, the rate limit assigned to a specific SL will limit only the traffic sent from the IB interface corresponding to the physical port GUID (virtual port0 IB interface). The traffic sent from other virtual IB interfaces will not be limited if no specific rules are defined.

Policy File Example

All physical ports in the fabric are with a rate limit of 50Mbps on SL1, except for GUID 0x2c90000000025, which is configured with rate limit of 25Mbps on SL1. In this example, the traffic on SLs (other than SL1) is unlimited.
All virtual ports in the fabric (except virtual port0 of all physical ports) will be rate-limited to 15Mbps for all SLs because of the default rule of VPORT_BW_RULES section.
Virtual port GUID 0x2c90000000026 is configured with a rate limit of 10Mbps on SL3. The rest of the SLs on this virtual port will get a rate limit of 15 Mbps because of the default rule of VPORT_BW_RULES section.

```
---------------------------------------------------------------------------
BW_NAMES
bw1 = 50
bw2 = 25
bw3 = 15
bw4 = 10

BW_RULES
default= 1:bw1
0x2c90000000025= 1:bw2

VPORT_BW_RULES
default= all:bw3
0x2c90000000026= 3:bw4

---------------------------------------------------------------------------
```

3.3.2.2.8 Adaptive Routing Manager and Self-Healing Networking

Adaptive Routing Manager supports advanced InfiniBand features; Adaptive Routing (AR) and Self-Healing Networking.

For information on how to set up AR and Self-Healing Networking, please refer to HowTo Configure Adaptive Routing and Self-Healing Networking Community post.

DOS MAD Prevention

DOS MAD prevention is achieved by assigning a threshold for each agent's RX. Agent's RX threshold provides a protection mechanism to the host memory by limiting the agents' RX with a threshold. Incoming MADs above the threshold are dropped and are not queued to the agent's RX.

To enable DOS MAD Prevention:

1. Go to /etc/modprobe.d/mlnx.conf.
2. Add to the file the option below.

```
ib_umad enable_rx_threshold 1
```

The threshold value can be controlled from the user-space via libibumad.

To change the value, use the following API:

```
int umad_update_threshold(int fd, int threshold);

@fd: file descriptor, agent's RX associated to this fd.
@threshold: new threshold value
```

3.3.2.2.9 IB Router Support in OpenSM

In order to enable the IB router in OpenSM, the following parameters should be configured:

IB Router Parameters for OpenSM

Parameter	Description	Default Value
`rtr_pr_flow_label`	Defines whether the SM should create alias GUIDs required for router support for each port. Defines flow label value to use in response for path records related to the router.	0 (Disabled)
rtr_pr_tclass	Defines TClass value to use in response for path records related to the router	0
rtr_pr_sl	Defines sl value to use in response for path records related to router.	0
rtr_p_mtu	Defines MTU value to use in response for path records related to the router.	4 (IB_MTU_LEN_2048)
rtr_pr_rate	Defines rate value to use in response for path records related to the router.	16 (IB_PATH_RECORD_RATE_100_GBS)

3.3.2.2.10 OpenSM Activity Report

OpenSM can produce an activity report in a form of a dump file which details the different activities done in the SM. Activities are divided into subjects. The OpenSM Supported Activities table below specifies the different activities currently supported in the SM activity report.
Reporting of each subject can be enabled individually using the configuration parameter `activity_report_subjects`:

- Valid values:
  Comma separated list of subjects to dump. The current supported subjects are:
  - "mc" - activity IDs 1, 2 and 8
  - "prtn" - activity IDs 3, 4, and 5
  - "virt" - activity IDs 6 and 7

- "routing" - activity IDs 8-12

Two predefined values can be configured as well:

- "all" - dump all subjects
- "none" - disable the feature by dumping none of the subjects

- Default value: "none"

OpenSM Supported Activities

ACtivity ID	Activity Name	Additional Fields	Comments	Description
1	mcm_member	• MLid • MGid • Port Guid • Join State	Join state: 1 - Join -1 - Leave	Member joined/ left MC group
2	mcg_change	• MLid • MGid • Change	Change: 0 - Create 1 - Delete	MC group created/deleted
3	prtn_guid_add	• Port Guid • PKey • Block index • Pkey Index		Guid added to partition
4	prtn_create	-PKey • Prtn Name		Partition created
5	prtn_delete	• PKey • Delete Reason	Delete Reason: 0 - empty prtn 1 - duplicate prtn 2 - sm shutdown	Partition deleted
6	port_virt_discover	• Port Guid • Top Index		Port virtualization discovered
7	vport_state_change	• Port Guid • VPort Guid • VPort Index • VNode Guid • VPort State	VPort State: 1 - Down 2 - Init 3 - ARMED 4 - Active	Vport state changed
8	mcg_tree_calc	mlid		MCast group tree calculated
9	routing_succeed	routing engine name		Routing done successfully
10	routing_failed	routing engine name		Routing failed
11	ucast_cache_invali-dated			ucast cache invalidated
12	ucast_cache_rout-ing_done			ucast cache routing done

3.3.2.2.11 Offsweep Balancing

When working with minhop/dor/updn, subnet manager can re-balance routing during idle time (between sweeps).

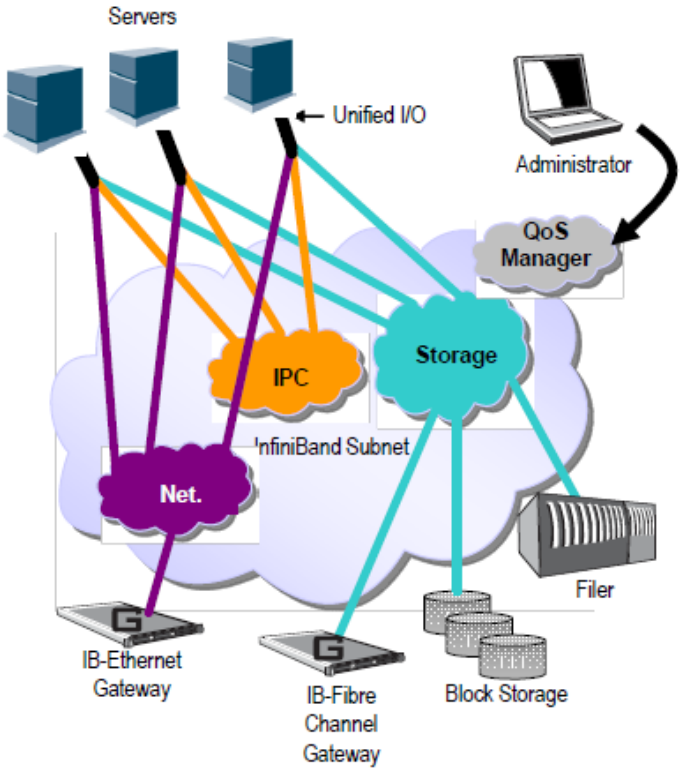
- offsweep_balancing_enabled - enables/disables the feature. Examples:
  - offsweep_balancing_enabled = TRUE

- offsweep_balancing_enabled = FALSE (default)

- offsweep_balancing_window - defines window of seconds to wait after sweep before starting the re-balance process. Applicable only if offsweep_balancing_enabled=TRUE. Example: offsweep_balancing_window = 180 (default)

3.3.2.3 QoS - Quality of Service

Quality of Service (QoS) requirements stem from the realization of I/O consolidation over an IB network. As multiple applications and ULPs share the same fabric, a means is needed to control their use of network resources.



The basic need is to differentiate the service levels provided to different traffic flows, such that a policy can be enforced and can control each flow utilization of fabric resources.
The InfiniBand Architecture Specification defines several hardware features and management interfaces for supporting QoS:

Up to 15 Virtual Lanes (VL) carry traffic in a non-blocking manner

- Arbitration between traffic of different VLs is performed by a two-priority-level weighted round robin arbiter. The arbiter is programmable with a sequence of (VL, weight) pairs and a maximal number of high priority credits to be processed before low priority is served
- Packets carry class of service marking in the range 0 to 15 in their header SL field
- Each switch can map the incoming packet by its SL to a particular output VL, based on a programmable table VL=SL-to-VL-MAP(in-port, out-port, SL)

- The Subnet Administrator controls the parameters of each communication flow by providing them as a response to Path Record (PR) or MultiPathRecord (MPR) queries

DiffServ architecture (IETF RFC 2474 & 2475) is widely used in highly dynamic fabrics. The following subsections provide the functional definition of the various software elements that enable a DiffServ-like architecture over the NVIDIA OFED software stack.

3.3.2.3.1 QoS Architecture

QoS functionality is split between the SM/SA, CMA and the various ULPs. We take the "chronology approach" to describe how the overall system works.

1. The network manager (human) provides a set of rules (policy) that define how the network is being configured and how its resources are split to different QoS-Levels. The policy also define how to decide which QoS-Level each application or ULP or service use.
2. The SM analyzes the provided policy to see if it is realizable and performs the necessary fabric setup. Part of this policy defines the default QoS-Level of each partition. The SA is enhanced to match the requested Source, Destination, QoS-Class, Service-ID, PKey against the policy, so clients (ULPs, programs) can obtain a policy enforced QoS. The SM may also set up partitions with appropriate IPoIB broadcast group. This broadcast group carries its QoS attributes: SL, MTU, RATE, and Packet Lifetime.
3. IPoIB is being setup. IPoIB uses the SL, MTU, RATE and Packet Lifetime available on the multicast group which forms the broadcast group of this partition.
4. MPI which provides non IB based connection management should be configured to run using hard coded SLs. It uses these SLs for every QP being opened.
5. ULPs that use CM interface (like SRP) have their own pre-assigned Service-ID and use it while obtaining PathRecord/MultiPathRecord (PR/MPR) for establishing connections. The SA receiving the PR/MPR matches it against the policy and returns the appropriate PR/MPR including SL, MTU, RATE and Lifetime.
6. ULPs and programs (e.g. SDP) use CMA to establish RC connection provide the CMA the target IP and port number. ULPs might also provide QoS-Class. The CMA then creates Service-ID for the ULP and passes this ID and optional QoS-Class in the PR/MPR request. The resulting PR/MPR is used for configuring the connection QP.

PathRecord and Multi Path Record Enhancement for QoS:

As mentioned above, the PathRecord and MultiPathRecord attributes are enhanced to carry the Service-ID which is a 64bit value. A new field QoS-Class is also provided.
A new capability bit describes the SM QoS support in the SA class port info. This approach provides an easy migration path for existing access layer and ULPs by not introducing new set of PR/MPR attributes.

3.3.2.3.2 Supported Policy

The QoS policy, which is specified in a stand-alone file, is divided into the following four subsections:

3.3.2.3.2.1 Port Group

A set of CAs, Routers or Switches that share the same settings. A port group might be a partition defined by the partition manager policy, list of GUIDs, or list of port names based on NodeDescription.

3.3.2.3.2.2 Fabric Setup

Defines how the SL2VL and VLArb tables should be set up.

> ⚠ In OFED this part of the policy is ignored. SL2VL and VLArb tables should be configured in the OpenSM options file (opensm.opts).

3.3.2.3.2.3 QoS-Levels Definition

This section defines the possible sets of parameters for QoS that a client might be mapped to. Each set holds SL and optionally: Max MTU, Max Rate, Packet Lifetime and Path Bits.

> ⚠ Path Bits are not implemented in OFED.

3.3.2.3.2.4 Matching Rules

A list of rules that match an incoming PR/MPR request to a QoS-Level. The rules are processed in order such as the first match is applied. Each rule is built out of a set of match expressions which should all match for the rule to apply. The matching expressions are defined for the following fields:

- SRC and DST to lists of port groups
- Service-ID to a list of Service-ID values or ranges
- QoS-Class to a list of QoS-Class values or ranges

3.3.2.3.3 CMA Features

The CMA interface supports Service-ID through the notion of port space as a prefix to the port number, which is part of the sockaddr provided to rdma_resolve_add(). The CMA also allows the ULP (like SDP) to propagate a request for a specific QoS-Class. The CMA uses the provided QoS-Class and Service-ID in the sent PR/MPR.

3.3.2.3.3.1 IPoIB

IPoIB queries the SA for its broadcast group information and uses the SL, MTU, RATE and Packet Lifetime available on the multicast group which forms this broadcast group.

3.3.2.3.3.2 SRP

The current SRP implementation uses its own CM callbacks (not CMA). So SRP fills in the Service-ID in the PR/MPR by itself and use that information in setting up the QP.
SRP Service-ID is defined by the SRP target I/O Controller (it also complies with IBTA Service- ID

rules). The Service-ID is reported by the I/O Controller in the ServiceEntries DMA attribute and should be used in the PR/MPR if the SA reports its ability to handle QoS PR/MPRs.

3.3.2.4 Secure Host

Secure host (supported in ConnectX®-3/ConnectX®-3 Pro adapter cards only) enables the device to protect itself and the subnet from malicious software. This is achieved by:

- Not allowing untrusted entities to access the device configuration registers, directly (through pci_cr or pci_conf) and indirectly (through MADs)
- Hiding the M_Key from the untrusted entities
- Preventing the modification of GUID0 by the untrusted entities
- Preventing drivers on untrusted hosts to receive or transmit SMP (QP0) MAD packets (SMP firewall)

When the SMP firewall is enabled, the firmware handles all QP0 packets, and does not forward them to the driver. Any information required by the driver for proper operation (e.g., SM lid) is passed via events generated by the firmware while processing QP0 MADs.
Driver support mainly requires using the MAD_DEMUX firmware command at driver startup.

3.3.2.4.1 Secure Mode Operation

Secure mode capability is enabled by setting the "cr_protection_en" parameter set to 1 in the [HCA] section of the .ini file and then burning the firmware with this .ini file. If the parameter is set to zero, or is missing, secure-mode operation will not be possible.
Once the firmware allows secure-mode operation, the secure-mode capability must be activated by using "flint" to set a 64-bit key (and then restarting the driver).
The flint command is as follows (the key is specified as up to 16 hex digits):

```
flint -d <device> set_key <64-bit key>
```

Example:

```
flint -d /dev/mst/mt26428_pci_cr0 set_key 1a1a1a1a2b2b2b2b
```

3.3.2.4.1.1 Enabling/Disabling Hardware Access

Once a 64-bit key is installed, the secure-mode is active once the driver is restarted. If the host is rebooted, the HCA comes out of reboot with secure-mode enabled. Hardware access can be disabled while the driver is running to enable operation such as maintenance, or firmware burning and then restored at the end of the operation.

The temporary enable does not affect the SMP firewall. This remains active even if the "cr-space" is temporarily permitted.

To enable hardware access:

```
flint -d /dev/mst/mt26428_pci_cr0 hw_access enable
Enter Key: ********
```

To disable hardware access:

```
flint -d /dev/mst/mt26428_pci_cr0 hw_access disable
```

If you do not explicitly restore hardware access when the maintenance operation is completed, the driver restart will NOT do so. The driver will come back after restart with hardware access disabled. Note, though, that the SMP firewall will remain active.

A host reboot will restore hardware access (with SMP firewall active). Thus, when you disable hardware access, you should restore it immediately after maintenance has been completed, either by using the flint command above or by rebooting the host (or both).

3.3.2.4.1.2 Burning New Firmware when Secure-Mode is Active

To burn a new firmware when the secure-mode is active:

1. Temporarily enable the hardware access (see Enabling/Disabling Hardware Access section).
2. Burn the new firmware.
3. Reboot the host (not just restart the driver).

3.3.2.4.1.3 Permanently Disabling Secure-Mode

To permanently disable secure-mode by setting the pass-key to zero:

1. Temporarily disable secure-mode (see Enabling/Disabling Hardware Access section).
2. Reset the pass-key to zero.

```
flint -d <device> set_key 0
```

3. Reboot the host.

This operation will cause the HCA to always come up (even from host reboot) in insecure mode. To restore security, simply set a non-zero pass-key again.

3.3.2.4.1.4 Checking if Hardware Access is Active

To check if hardware access is active:

```
flint -d /dev/mst/mt26428_pci_cr0 -qq q
```

If the hardware access is active, you will see the following error message:

```
E- Cannot open /dev/mst/mt26428_pci_cr0: HW access is disabled on the device.
E- Run "flint -d /dev/mst/mt26428_pci_cr0 hw_access enable" in order to enable HW access.
```

3.3.2.4.1.5 Checking if the SMP Firewall is Active

The SMP firewall is active as long as there is a non-zero pass-key active in the firmware (regardless of whether or not the Secure-Mode has been temporarily disabled).
To check if SMP Firewall is active, run the InfiniBand diagnostic command sminfo.
If the SMP firewall is active, the command will fail as shown below:

```
[root@dev-l-vrt-016 ~]# sminfo
ibwarn: [26872] mad_rpc: _do_madrpc failed; dport (Lid 1)
sminfo: iberror: failed: query
```

3.3.2.5 IP over InfiniBand (IPoIB)

3.3.2.5.1 Upper Layer Protocol (ULP)

The IP over IB (IPoIB) ULP driver is a network interface implementation over InfiniBand. IPoIB encapsulates IP datagrams over an InfiniBand Connected or Datagram transport service. The IPoIB driver, ib_ipoib, exploits the following capabilities:

- VLAN simulation over an InfiniBand network via child interfaces
- High Availability via Bonding
- Varies MTU values:
    - up to 4k in Datagram mode
- Uses any ConnectX® IB ports (one or two)
- Inserts IP/UDP/TCP checksum on outgoing packets
- Calculates checksum on received packets
- Support net device TSO through ConnectX® LSO capability to defragment large data- grams to MTU quantas.

IPoIB also supports the following software based enhancements:

- Giant Receive Offload
- NAPI
- Ethtool support

3.3.2.5.2 Enhanced IPoIB

Enhanced IPoIB feature enables offloading ULP basic capabilities to a lower vendor specific driver, in order to optimize IPoIB data path. This will allow IPoIB to support multiple stateless offloads, such as RSS/TSS, and better utilize the features supported, enabling IPoIB datagram to reach peak performance in both bandwidth and latency.

Enhanced IPoIB supports/performs the following:

- Stateless offloads (RSS, TSS)
- Multi queues
- Interrupt moderation
- Multi partitions optimizations
- Sharing send/receive Work Queues
- Vendor specific optimizations
- UD mode only

3.3.2.5.3 Port Configuration

The physical port MTU (indicates the port capability) default value is 4k, whereas the IPoIB port MTU ("logical" MTU) default value is 2k as it is set by the OpenSM.
To change the IPoIB MTU to 4k, edit the OpenSM partition file in the section of IPoIB setting as follow:

```
Default=0xffff, ipoib, mtu=5 : ALL=full;
```

where:

"mtu=5" indicates that all IPoIB ports in the fabric are using 4k MTU, ("mtu=4" indi- cates 2k MTU)

3.3.2.5.4 IPoIB Configuration

Unless you have run the installation script mlnxofedinstall with the flag '-n', then IPoIB has not been configured by the installation. The configuration of IPoIB requires assigning an IP address and a subnet mask to each HCA port, like any other network adapter card (i.e., you need to prepare a file called ifcfg-ib<n> for each port). The first port on the first HCA in the host is called interface ib0, the second port is called ib1, and so on.
IPoIB configuration can be based on DHCP or on a static configuration that you need to supply (see below). You can also apply a manual configuration that persists only until the next reboot or driver restart (see below).

3.3.2.5.4.1 IPoIB Configuration Based on DHCP

Setting an IPoIB interface configuration based on DHCP is performed similarly to the configuration of Ethernet interfaces. In other words, you need to make sure that IPoIB configuration files include the following line:

- For RedHat:

```
BOOTPROTO=dhcp
```

- For SLES:

```
BOOTPROTO='dchp'
```

> If IPoIB configuration files are included, ifcfg-ib<n> files will be installed under:
>
> /etc/sysconfig/network-scripts/ on a RedHat machine
> /etc/sysconfig/network/ on a SuSE machine.

> A patch for DHCP may be required for supporting IPoIB. For further information, please see the REAME file available under the docs/dhcp/ directory.

> Red Hat Enterprise Linux 7 supports assigning static IP addresses to InfiniBand IPoIB interfaces. However, as these interfaces do not have a normal hardware Ethernet address, a different method of specifying a unique identifier for the IPoIB interface must be used. The standard is to use the option dhcp-client-identifier= construct to specify the IPoIB interface's dhcp-client-identifier field. The DHCP server host construct supports at most one hardware Ethernet and one dhcp-client-identifier entry per host stanza. However, there may be more than one fixed-address entry and the DHCP server will automatically respond with an address that is appropriate for the network that the DHCP request was received on.

Standard DHCP fields holding MAC addresses are not large enough to contain an IPoIB hardware address. To overcome this problem, DHCP over InfiniBand messages convey a client identifier field used to identify the DHCP session. This client identifier field can be used to associate an IP address with a client identifier value, such that the DHCP server will grant the same IP address to any client that conveys this client identifier.
The length of the client identifier field is not fixed in the specification. For the *NVIDIA OFED for Linux* package, it is recommended to have IPoIB use the same format that FlexBoot uses for this client identifier.

3.3.2.5.4.2 DHCP Server

In order for the DHCP server to provide configuration records for clients, an appropriate configuration file needs to be created. By default, the DHCP server looks for a configuration file called dhcpd.conf under /etc. You can either edit this file or create a new one and provide its full path to the DHCP server using the -cf flag (See a file example at docs/dhcpd.conf).
The DHCP server must run on a machine which has loaded the IPoIB module. To run the DHCP server from the command line, enter:

```
dhcpd <IB network interface name> -d
```

Example:

```
host1# dhcpd ib0 -d
```

3.3.2.5.4.3 DHCP Client (Optional)

⚠ A DHCP client can be used if you need to prepare a diskless machine with an IB driver.

In order to use a DHCP client identifier, you need to first create a configuration file that defines the DHCP client identifier.

Then run the DHCP client with this file using the following command:

```
dhclient -cf <client conf file> <IB network interface name>
```

Example of a configuration file for the ConnectX (PCI Device ID 26428), called `dhclient.conf`:

```
The value indicates a hexadecimal number interface "ib1" {
send dhcp-client-identifier
ff:00:00:00:00:02:00:00:02:c9:00:00:02:c9:03:00:00:10:39;
}
```

Example of a configuration file for InfiniHost III Ex (PCI Device ID 25218), called `dhclient.conf`:

```
The value indicates a hexadecimal number interface "ib1" {
send dhcp-client-identifier
20:00:55:04:01:fe:80:00:00:00:00:00:00:02:c9:02:00:23:13:92;
}
```

➤ *In order to use the configuration file, run*:

```
host1# dhclient -cf dhclient.conf ib1
```

3.3.2.5.4.4 Static IPoIB Configuration

If you wish to use an IPoIB configuration that is not based on DHCP, you need to supply the installation script with a configuration file (using the '-n' option) containing the full IP configuration. The IPoIB configuration file can specify either or both of the following data for an IPoIB interface:

- A static IPoIB configuration
- An IPoIB configuration based on an Ethernet configuration
  See your Linux distribution documentation for additional information about configuring IP addresses.

The following code lines are an excerpt from a sample IPoIB configuration file:

```
# Static settings; all values provided by this file
IPADDR_ib0=10.4.3.175
NETMASK_ib0=255.255.0.0
NETWORK_ib0=10.4.0.0
BROADCAST_ib0=10.4.255.255
ONBOOT_ib0=1
# Based on eth0; each '*' will be replaced with a corresponding octet
```

```
# from eth0.
LAN_INTERFACE_ib0=eth0
IPADDR_ib0=10.4.'*'.'*'
NETMASK_ib0=255.255.0.0
NETWORK_ib0=10.4.0.0
BROADCAST_ib0=10.4.255.255
ONBOOT_ib0=1
# Based on the first eth<n> interface that is found (for n=0,1,...);
# each '*' will be replaced with a corresponding octet from eth<n>.
LAN_INTERFACE_ib0=
IPADDR_ib0=10.4.'*'.'*'
NETMASK_ib0=255.255.0.0
NETWORK_ib0=10.4.0.0
BROADCAST_ib0=10.4.255.255
ONBOOT_ib0=1
```

3.3.2.5.4.5 Manually Configuring IPoIB

⚠ This manual configuration persists only until the next reboot or driver restart.

To manually configure IPoIB for the default IB partition (VLAN), perform the following steps:

1. Configure the interface by entering the ifconfig command with the following items:
   - The appropriate IB interface (ib0, ib1, etc.)
   - The IP address that you want to assign to the interface
   - The netmask keyword
   - The subnet mask that you want to assign to the interface

   The following example shows how to configure an IB interface:

   ```
   host1$ ifconfig ib0 10.4.3.175 netmask 255.255.0.0
   ```

2. (Optional) Verify the configuration by entering the ifconfig command with the appropriate interface identifier *ib#* argument.
   The following example shows how to verify the configuration:

   ```
   host1$ ifconfig ib0
   b0  Link encap:UNSPEC  HWaddr 80-00-04-04-FE-80-00-00-00-00-00-00-00-00-00-00
   inet addr:10.4.3.175  Bcast:10.4.255.255  Mask:255.255.0.0
   UP BROADCAST MULTICAST  MTU:65520  Metric:1
   RX packets:0 errors:0 dropped:0 overruns:0 frame:0
   TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
   collisions:0 txqueuelen:128
   RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
   ```

3. Repeat the first two steps on the remaining interface(s).

3.3.2.5.5 Sub-interfaces

You can create sub-interfaces for a primary IPoIB interface to provide traffic isolation. Each such sub-interface (also called a child interface) has a different IP and network addresses from the primary (parent) interface. The default Partition Key (PKey), ff:ff, applies to the primary (parent) interface.
This section describes how to:

- Create a subinterface
- Remove a subinterface

3.3.2.5.5.1 Creating a Subinterface

In the following procedure, ib0 is used as an example of an IB sub-interface.

To create a child interface (sub-interface), follow this procedure:

1. Decide on the PKey to be used in the subnet (valid values can be 0 or any 16-bit unsigned value). The actual PKey used is a 16-bit number with the most significant bit set. For example, a value of 1 will give a PKey with the value 0x8001.
2. Create a child interface by running:

   ```
   host1$ echo <PKey> > /sys/class/net/<IB subinterface>/create_child
   ```

   Example:

   ```
   host1$ echo 1 > /sys/class/net/ib0/create_child
   ```

   This will create the interface ib0.8001.
3. Verify the configuration of this interface by running:

   ```
   host1$ ifconfig <subinterface>.<subinterface PKey>
   ```

   Using the example of the previous step:

   ```
   host1$ ifconfig ib0.8001
   ib0.8001  Link encap:UNSPEC  HWaddr 80-00-00-4A-FE-80-00-00-00-00-00-00-00-00-00-00
   BROADCAST MULTICAST  MTU:2044  Metric:1
   RX packets:0 errors:0 dropped:0 overruns:0 frame:0
   TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
   collisions:0 txqueuelen:128
   RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
   ```
4. As can be seen, the interface does not have IP or network addresses. To configure those, you should follow the manual configuration procedure described in "Manually Configuring IPoIB" section above.
5. To be able to use this interface, a configuration of the Subnet Manager is needed so that the PKey chosen, which defines a broadcast address, be recognized.

3.3.2.5.5.2 Removing a Subinterface

To remove a child interface (subinterface), run:

```
echo <subinterface PKey> /sys/class/net/<ib_interface>/delete_child
```

Using the example of the second step from the previous chapter:

```
echo 0x8001 > /sys/class/net/ib0/delete_child
```

Note that when deleting the interface you must use the PKey value with the most significant bit set (e.g., 0x8000 in the example above).

3.3.2.5.6 Verifying IPoIB Functionality

To verify your configuration and IPoIB functionality are successful, perform the following steps:

1. Verify the IPoIB functionality by using the `ifconfig` command.
   The following example shows how two IB nodes are used to verify IPoIB functionality. In the following example, IB node 1 is at 10.4.3.175, and IB node 2 is at 10.4.3.176:

   ```
   host1# ifconfig ib0 10.4.3.175 netmask 255.255.0.0
   host2# ifconfig ib0 10.4.3.176 netmask 255.255.0.0
   ```

2. Enter the ping command from 10.4.3.175 to 10.4.3.176.
3. The following example shows how to enter the ping command:

   ```
   host1# ping -c 5 10.4.3.176
   PING 10.4.3.176 (10.4.3.176) 56(84) bytes of data.
   64 bytes from 10.4.3.176: icmp_seq=0 ttl=64 time=0.079 ms
   64 bytes from 10.4.3.176: icmp_seq=1 ttl=64 time=0.044 ms
   64 bytes from 10.4.3.176: icmp_seq=2 ttl=64 time=0.055 ms
   64 bytes from 10.4.3.176: icmp_seq=3 ttl=64 time=0.049 ms
   64 bytes from 10.4.3.176: icmp_seq=4 ttl=64 time=0.065 ms
   --- 10.4.3.176 ping statistics ---
   5 packets transmitted, 5 received, 0% packet loss, time 3999ms rtt min/avg/max/mdev = 0.044/0.058/0.079/
   0.014 ms, pipe 2
   ```

3.3.2.5.7 Bonding IPoIB

To create an interface configuration script for the ibX and bondX interfaces, you should use the standard syntax (depending on your OS).
Bonding of IPoIB interfaces is accomplished in the same manner as would bonding of Ethernet interfaces: via the Linux Bonding Driver.

- Network Script files for IPoIB slaves are named after the IPoIB interfaces (e.g: ifcfg- ib0)
- The only meaningful bonding policy in IPoIB is High-Availability (bonding mode number 1, or active-backup)
- Bonding parameter "fail_over_mac" is meaningless in IPoIB interfaces, hence, the only supported value is the default: 0

For a persistent bonding IPoIB Network configuration, use the same Linux Network Scripts semantics, with the following exceptions/ additions:

- In the bonding master configuration file (e.g: ifcfg-bond0), in addition to Linux bonding semantics, use the following parameter: MTU=65520
  COND

  For IPoIB slaves, use MTU=2044. If you do not set the correct MTU or do not set MTU at all, performance of the interface might decrease.

  Dynamically Connected Transport (DCT)

- In the bonding slave configuration file (e.g: ifcfg-ib0), use the same Linux Network Scripts semantics. In particular: DEVICE=ib0
- In the bonding slave configuration file (e.g: ifcfg-ib0.8003), the line TYPE=InfiniBand is necessary when using bonding over devices configured with partitions (p_key)
- For RHEL users:
  In /etc/modprobe.b/bond.conf add the following lines:

  ```
  alias bond0 bonding
  ```

- For SLES users:
  It is necessary to update the MANDATORY_DEVICES environment variable in /etc/sysconfig/network/config with the names of the IPoIB slave devices (e.g. ib0, ib1, etc.). Otherwise, bonding master may be created before IPoIB slave interfaces at boot time.
  It is possible to have multiple IPoIB bonding masters and a mix of IPoIB bonding master and Ethernet bonding master. However, It is NOT possible to mix Ethernet and IPoIB slaves under the same bonding master.

Restarting openibd does no keep the bonding configuration via Network Scripts. You have to restart the network service in order to bring up the bonding master. After the configuration is saved, restart the network service by running: /etc/init.d/network restart.

3.3.2.5.8 Dynamic PKey Change

Dynamic PKey change means the PKey can be changed (add/removed) in the SM database and the interface that is attached to that PKey is updated immediately without the need to restart the driver.
If the PKey is already configured in the port by the SM, the child-interface can be used immediately. If not, the interface will be ready to use only when SM adds the relevant PKey value to the port after the creation of the child interface. No additional configuration is required once the child-interface is created.

3.3.2.5.9 Precision Time Protocol (PTP) over IPoIB

This feature allows for accurate synchronization between the distributed entities over the network. The synchronization is based on symmetric Round Trip Time (RTT) between the master and slave devices.

This feature is enabled by default, and is also supported over PKey interfaces.

For more on the PTP feature, refer to Running Linux PTP with ConnectX-4/ConnectX-5/ConnectX-6 Community post.

For further information on Time-Stamping, follow the steps in "Time-Stamping Service".

3.3.2.5.10 One Pulse Per Second (1PPS) over IPoIB

1PPS is a time synchronization feature that allows the adapter to be able to send or receive 1 pulse per second on a dedicated pin on the adapter card using an SMA connector (SubMiniature version A). Only one pin is supported and could be configured as 1PPS in or 1PPS out.
For further information, refer to HowTo Test 1PPS on NVIDIA Adapters Community post.

3.3.2.6 Advanced Transport

3.3.2.6.1 Atomic Operations

3.3.2.6.1.1 Atomic Operations in mlx5 Driver

To enable atomic operation with this endianness contradiction, use the `ibv_create_qp` to create the QP and set the `IBV_QP_CREATE_ATOMIC_BE_REPLY` flag on `create_flags`.

3.3.2.6.2 XRC - eXtended Reliable Connected Transport Service for InfiniBand

XRC allows significant savings in the number of QPs and the associated memory resources required to establish all to all process connectivity in large clusters.
It significantly improves the scalability of the solution for large clusters of multicore end-nodes by reducing the required resources.
For further details, please refer to the "Annex A14 Supplement to InfiniBand Architecture Specification Volume 1.2.1"
A new API can be used by user space applications to work with the XRC transport. The legacy API is currently supported in both binary and source modes, however it is deprecated. Thus we recommend using the new API.
The new verbs to be used are:

- ibv_open_xrcd/ibv_close_xrcd
- ibv_create_srq_ex
- ibv_get_srq_num
- ibv_create_qp_ex
- ibv_open_qp

Please use `ibv_xsrq_pingpong` for basic tests and code reference. For detailed information regarding the various options for these verbs, please refer to their appropriate man pages.

3.3.2.6.3 Dynamically Connected Transport (DCT)

Dynamically Connected transport (DCT) service is an extension to transport services to enable a higher degree of scalability while maintaining high performance for sparse traffic. Utilization of DCT reduces the total number of QPs required system wide by having Reliable type QPs dynamically connect and disconnect from any remote node. DCT connections only stay connected while they are

active. This results in smaller memory footprint, less overhead to set connections and higher on-chip cache utilization and hence increased performance. DCT is supported only in mlx5 driver.

> ⚠ Please note that ConnectX-4 supports DCT v0 and ConnectX-5 and above support DCT v1. DCTv0 and DCT v1 are not interoperable.

3.3.2.6.4 MPI Tag Matching and Rendezvous Offloads

> ⚠ Supported in ConnectX®-5 and above adapter cards.

Tag Matching and Rendezvous Offloads is a technology employed by NVIDIA to offload the processing of MPI messages from the host machine onto the network card. Employing this technology enables a zero copy of MPI messages, i.e. messages are scattered directly to the user's buffer without intermediate buffering and copies. It also provides a complete rendezvous progress by NVIDIA devices. Such overlap capability enables the CPU to perform the application's computational tasks while the remote data is gathered by the adapter.
For more information Tag Matching Offload, please refer to the Understanding MPI Tag Matching and Rendezvous Offloads (ConnectX-5) Community post.

3.3.2.7 Optimized Memory Access

3.3.2.7.1 Memory Region Re-registration

Memory Region Re-registration allows the user to change attributes of the memory region. The user may change the PD, access flags or the address and length of the memory region. Memory region supports contagious pages allocation. Consequently, it de-registers memory region followed by register memory region. Where possible, resources are reused instead of de-allocated and reallocated.

> ⚠ Please note that the verb is implemented as an experimental verb.

Example:

```
int ibv_rereg_mr(struct ibv_mr *mr, int flags, struct ibv_pd *pd, void *addr, size_t length, uint64_t access,
struct ibv_rereg_mr_attr *attr);
```

@mr:	The memory region to modify.
@flags:	A bit-mask used to indicate which of the following properties of the memory region are being modified. Flags should be one of: IBV_REREG_MR_CHANGE_TRANSLATION /* Change translation (location and length) */ IBV_REREG_MR_CHANGE_PD/* Change protection domain*/ IBV_REREG_MR_CHANGE_ACCESS/* Change access flags*/
@pd:	If IBV_REREG_MR_CHANGE_PD is set in flags, this field specifies the new protection domain to associated with the memory region, otherwise, this parameter is ignored.

@addr:	If IBV_REREG_MR_CHANGE_TRANSLATION is set in flags, this field specifies the start of the virtual address to use in the new translation, otherwise, this parameter is ignored.
@length:	If IBV_REREG_MR_CHANGE_TRANSLATION is set in flags, this field specifies the length of the virtual address to use in the new translation, otherwise, this parameter is ignored.
@access:	If IBV_REREG_MR_CHANGE_ACCESS is set in flags, this field specifies the new memory access rights, otherwise, this parameter is ignored. Could be one of the following: IBV_ACCESS_LOCAL_WRITE IBV_ACCESS_REMOTE_WRITE IBV_ACCESS_REMOTE_READ IBV_ACCESS_ALLOCATE_MR /* Let the library allocate the memory for * the user, tries to get contiguous pages */
@attr:	Future extensions

ibv_rereg_mr returns 0 on success, or the value of an errno on failure (which indicates the error reason). In case of an error, the MR is in undefined state. The user needs to call ibv_dereg_mr in order to release it.

Please note that if the MR (Memory Region) is created as a Shared MR and a translation is requested, after the call, the MR is no longer a shared MR. Moreover, Re-registration of MRs that uses NVIDIA PeerDirect™ technology are not supported.

3.3.2.7.2 Memory Window

Memory Window allows the application to have a more flexible control over remote access to its memory. It is available only on physical functions/native machines The two types of Memory Windows supported are: type 1 and type 2B.
Memory Windows are intended for situations where the application wants to:

- Grant and revoke remote access rights to a registered region in a dynamic fashion with less of a performance penalty
- Grant different remote access rights to different remote agents and/or grant those rights over different ranges within registered region

For further information, please refer to the InfiniBand specification document.

Memory Windows API cannot co-work with peer memory clients (PeerDirect).

3.3.2.7.2.1 Query Capabilities

Memory Windows are available if and only the hardware supports it. To verify whether Memory Windows are available, run `ibv_query_device`.
For example:

```
truct ibv_device_attr device_attr = {.comp_mask = IBV_DEVICE_ATTR_RESERVED - 1};
ibv_query_device(context, & device_attr);
if (device_attr.exp_device_cap_flags & IBV_DEVICE_MEM_WINDOW ||
                device_attr.exp_device_cap_flags & IBV_DEVICE_MW_TYPE_2B) {
/* Memory window is supported */
```

3.3.2.7.2.2 Memory Window Allocation

Allocating memory window is done by calling the `ibv_alloc_mw` verb.

```
type_mw = IBV_MW_TYPE_2/ IBV_MW_TYPE_1
mw = ibv_alloc_mw(pd, type_mw);
```

3.3.2.7.2.3 Binding Memory Windows

After being allocated, memory window should be bound to a registered memory region. Memory Region should have been registered using the IBV_ACCESS_MW_BIND access flag.

For further information on how to bind memory windows, please see rdma-core man page.

3.3.2.7.2.4 Invalidating Memory Window

Before rebinding Memory Window type 2, it must be invalidated using `ibv_post_send` - see here.

3.3.2.7.2.5 Deallocating Memory Window

Deallocating memory window is done using the ibv_dealloc_mw verb.

```
ibv_dealloc_mw(mw);
```

3.3.2.7.3 User-Mode Memory Registration (UMR)

User-mode Memory Registration (UMR) is a fast registration mode which uses send queue. The UMR support enables the usage of RDMA operations and scatters the data at the remote side through the definition of appropriate memory keys on the remote side.
UMR enables the user to:

- Create indirect memory keys from previously registered memory regions, including creation of KLM's from previous KLM's. There are not data alignment or length restrictions associated with the memory regions used to define the new KLM's.
- Create memory regions, which support the definition of regular non-contiguous memory regions.

3.3.2.7.4 On-Demand-Paging (ODP)

On-Demand-Paging (ODP) is a technique to alleviate much of the shortcomings of memory registration. Applications no longer need to pin down the underlying physical pages of the address space, and track the validity of the mappings. Rather, the HCA requests the latest translations from the OS when pages are not present, and the OS invalidates translations which are no longer valid due to either non-present pages or mapping changes. ODP does not support contiguous pages.
ODP can be further divided into 2 subclasses: Explicit and Implicit ODP.

- Explicit ODP
  In Explicit ODP, applications still register memory buffers for communication, but this operation is used to define access control for IO rather than pin-down the pages. ODP Memory Region (MR) does not need to have valid mappings at registration time.

- Implicit ODP
  In Implicit ODP, applications are provided with a special memory key that represents their complete address space. This all IO accesses referencing this key (subject to the access rights associated with the key) does not need to register any virtual address range.

3.3.2.7.4.1 Query Capabilities

On-Demand Paging is available if both the hardware and the kernel support it. To verify whether ODP is supported, run `ibv_query_device.`

For further information, please refer to the `ibv_query_device` manual page.

3.3.2.7.4.2 Registering ODP Explicit and Implicit MR

ODP Explicit MR is registered after allocating the necessary resources (e.g. PD, buffer), while ODP implicit MR registration provides an implicit lkey that represents the complete address space.

For further information, please refer to the `ibv_reg_mr` manual page.

3.3.2.7.4.3 De-registering ODP MR

ODP MR is deregistered the same way a regular MR is deregistered:

```
ibv_dereg_mr(mr);
```

3.3.2.7.4.4 Advice MR Verb

The driver can pre-fetch a given range of pages and map them for access from the HCA. The advice MR verb is applicable for ODP MRs only.
For further information, please refer to the `ibv_advise_mr` manual page.

3.3.2.7.4.5 ODP Statistics

To aid in debugging and performance measurements and tuning, ODP support includes an extensive set of statistics.

For further information, please refer to rdma-statistics manual page.

3.3.2.7.5 Inline-Receive

The HCA may write received data to the Receive CQE. Inline-Receive saves PCIe Read transaction since the HCA does not need to read the scatter list. Therefore, it improves performance in case of short receive-messages.

On poll CQ, the driver copies the received data from CQE to the user's buffers.

Inline-Receive is enabled by default and is transparent to the user application. To disable it globally, set MLX5_SCATTER_TO_CQE environment variable to the value of 0. Otherwise, disable it on a specific QP using mlx5dv_create_qp() with MLX5DV_QP_CREATE_DISABLE_SCATTER_TO_CQE.

For further information, please refer to the manual page of mlx5dv_create_qp().

3.3.2.8 NVIDIA PeerDirect

NVIDIA PeerDirect™ uses an API between IB CORE and peer memory clients, (e.g. GPU cards) to provide access to an HCA to read/write peer memory for data buffers. As a result, it allows RDMA-based (over InfiniBand/RoCE) application to use peer device computing power, and RDMA interconnect at the same time without copying the data between the P2P devices.

For example, PeerDirect is being used for GPUDirect RDMA.

Detailed description for that API exists under MLNX OFED installation, please see `docs/readme_and_user_manual/PEER_MEMORY_API.txt`.

3.3.2.8.1 PeerDirect Async

Mellanox PeerDirect Async sub-system gives PeerDirect hardware devices, such as GPU cards, dedicated AS accelerators, and so on, the ability to take control over HCA in critical path offloading CPU. To achieve this, there is a set of verb calls and structures providing application with abstract description of operation sequences intended to be executed by peer device.

3.3.2.8.2 Relaxed Ordering (RSYNC)

This feature is only supported on ConnectX-5 adapter cards and above.

In GPU systems with relaxed ordering, RSYNC callback will be invoked to ensure memory consistency. The registration and implementation of the callback will be done using an external module provided by the system vendor. Loading the module will register the callback in MLNX_OFED to be used later to guarantee memory operations order.

3.3.2.9 CPU Overhead Distribution

When creating a CQ using the `ibv_create_cq()` API, a "`comp_vector`" argument is sent. If the value set for this argument is 0, while the CPU core executing this verb is not equal to zero, the driver assigns a completion EQ with the least CQs reporting to it. This method is used to distribute CQs amongst available completions EQ. To assign a CQ to a specific EQ, the EQ needs to be specified in the `comp_vector` argument.

3.3.2.10 Resource Domain Experimental Verbs

Resource domain is a verb object that may be associated with QP and CQ objects upon creation to enhance data-path performance.

3.3.2.10.1 Resource Domain Attributes

Thread model - defines the threading module for the objects (QP/CQ) associated with this resource domain.

`IBV_EXP_THREAD_SAFE`	Access to the associated objects are thread safe. Such objects may be accessed by any thread without concern for thread safety issues.

`IBV_EXP_THREAD_UNSAFE`	Access to the associated objects are not thread safe. Access to such objects must be coordinated by the calling threads.
`IBV_EXP_THREAD_SINGLE`	Access to the associated objects are from only one thread over the lifetime of the object. In addition, different objects associated with the same resource domain must be called by the same thread.

Message model - defines whether associated objects connection is optimized for low latency or high bandwidth.

`IBV_EXP_MSG_FORCE_LOW_LATENCY`	Forces the provider to optimize for low latency
`IBV_EXP_MSG_LOW_LATENCY`	Suggests the provider to optimize for low latency.
`IBV_EXP_MSG_HIGH_BW`	Suggests the provider to optimize for high bandwidth.
`IBV_EXP_MSG_DEFAULT`	Uses the provider's default message model.

To create a resource domain, use:

```
static inline struct ibv_exp_res_domain *ibv_exp_create_res_domain(
struct ibv_context *context, struct ibv_exp_res_domain_init_attr *attr)
```

To destroy a resource domain, use:

```
static inline int ibv_exp_destroy_res_domain( struct ibv_context *context,
struct ibv_exp_res_domain *res_dom, struct ibv_exp_destroy_res_domain_attr *attr)
```

Use the `res_domain` field and relevant `comp_mask` in the `ibv_exp_cq_init_attr` and `ibv_exp_qp_init_attr` structs to pass resource domain to the CQ and QP upon their creation.

Example - creating a CQ and QP that may be called from one thread **only:**

```
struct ibv_exp_res_domain_init_attr res_domain_attr;
struct ibv_exp_res_domain *res_domain;
struct ibv_exp_cq_init_attr cq_attr;
struct ibv_exp_qp_init_attr qp_attr;

…
res_domain_attr.comp_mask = IBV_EXP_RES_DOMAIN_THREAD_MODEL |
IBV_EXP_RES_DOMAIN_MSG_MODEL;
res_domain_attr.thread_model = IBV_EXP_THREAD_SINGLE;
res_domain_attr.msg_model = IBV_EXP_MSG_HIGH_BW;

res_domain = ibv_exp_create_res_domain(ctx->context, &res_domain_attr);
if (!_domain) {
     fprintf(stderr, "Can't create resource domain\n");
     exit(1);
}
…
cq_attr.res_domain = res_domain;
cq_attr.comp_mask |= IBV_EXP_CQ_INIT_ATTR_RES_DOMAIN;
```

```
cq = ibv_exp_create_cq(context, num_ entries, NULL, NULL, 0, &cq_attr);
…
qp_attr.res_domain = res_domain;
qp_attr.comp_mask |= IBV_EXP_QP_INIT_ATTR_RES_DOMAIN;
qp = ibv_exp_create_qp(context, &qp_attr);
```

3.3.2.11 Query Interface Experimental Verbs

The new experimental verbs `ibv_exp_query_intf` and `ibv_exp_release_intf` provide a mechanism to extend the verbs with families of verbs interfaces. These extensions provide a way to optimize data-path interfaces (e.g. `post-send/recv, poll-cq`) for specific applications (e.g. DPDK).
The interfaces families provided by the new verbs may be vendor specific families (in this case, the vendor should provide the header file for the interface definition) or global families which will be defined in `verbs.h`.
For more information regarding the new verbs, please refer to their man pages and the `ibv_intf` example.

3.3.2.11.1 QP-burst Experimental Family

The `ibv_exp_qp_burst_family` is an extension interface to the legacy post_send/receive verbs. This family provides an interface for applications that need fast `send/recv` messages (e.g. DPDK).
To get an instance of `ibv_exp_qp_burst_family`, the application needs to use the query-interface mechanism (`ibv_exp_query_intf`).
For more information on the `ibv_exp_qp_burst_family`, please refer to its man page and the `ibv_intf` example.

3.3.2.11.2 CQ Experimental Family

The `ibv_exp_cq_family` is an extension interface to the legacy `poll_cq` verb. This family provides an interface for applications that need fast `send/recv` messages (e.g. DPDK).
To get an instance of the `ibv_exp_cq_family` the application needs to use the query-interface mechanism (`ibv_exp_query_intf`).
For more information regarding the `ibv_exp_cq_family`, please refer to its man page and the `ibv_intf` example.

3.3.2.12 Out-of-Order (OOO) Data Placement

⚠ This feature is only supported on:

- ConnectX-5 adapter cards and above
- RC and XRC QPs
- DC transport

3.3.2.12.1 Overview

In certain fabric configurations, InfiniBand packets for a given QP may take up different paths in a network from source to destination. This results into packets being received in an out-of-order

manner. These packets can now be handled instead of being dropped, in order to avoid retransmission, by:

- Achieving better network utilization
- Decreasing latency

Data will be placed into host memory in an out-of-order manner when out-of-order messages are received.

For information on how to set up out-of-order processing by the QP, please refer to HowTo Configure Adaptive Routing and SHIELD Community post.

3.3.2.13 WQE Format in MLNX_OFED

Please refer to "Wake-on-LAN (WoL)" section.

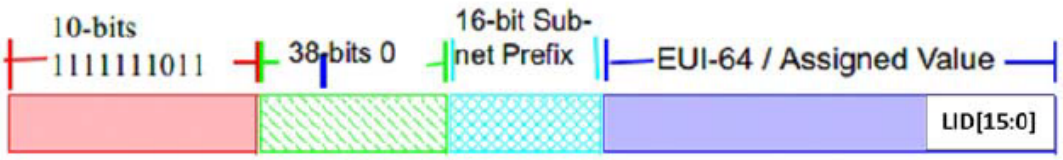
3.3.2.14 IB Router

IB router provides the ability to send traffic between two or more IB subnets thereby potentially expanding the size of the network to over 40k end-ports, enabling separation and fault resilience between islands and IB subnets, and enabling connection to different topologies used by different subnets.
The forwarding between the IB subnets is performed using GRH lookup. The IB router's basic functionality includes:

- Removal of current L2 LRH (local routing header)
- Routing
- table lookup – using GID from GRH
- Building new LRH according to the destination according to the routing table

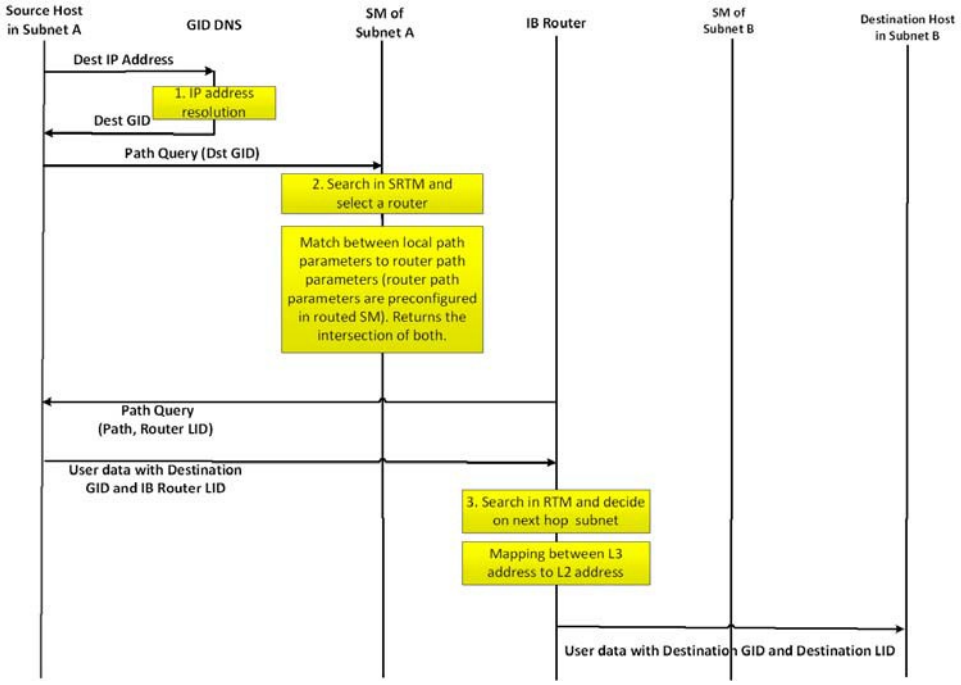
The DLID in the new LRH is built using simplified GID-to-LID mapping (where LID = 16 LSB bits of GID) thereby not requiring to send for ARP query/lookup.

Local Unicast GID Format



For this to work, the SM allocates an alias GID for each host in the fabric where the alias GID = {subnet prefix[127:64], reserved[63:16], LID[15:0}. Hosts should use alias GIDs in order to transmit traffic to peers on remote subnets.

Host-to-Host IB Router Unicast Flow



- For information on the architecture and functionality of IB Router, refer to IB Router Architecture and Functionality Community post.
- For information on IB Router configuration, refer to HowTo Configure IB Routers Community post.

3.3.2.15 MAD Congestion Control

The SA Management Datagrams (MAD) are General Management Packets (GMP) used to communicate with the SA entity within the InfiniBand subnet. SA is normally part of the subnet manager, and it is contained within a single active instance. Therefore, congestion on the SA communication level may occur.

Congestion control is done by allowing max_outstanding MADs only, where outstanding MAD means that is has no response yet. It also holds a FIFO queue that holds the SA MADs that their sending is delayed due to max_outstanding overflow.

The length of the queue is queue_size and meant to limit the FIFO growth beyond the machine memory capabilities. When the FIFO is full, SA MADs will be dropped, and the drops counter will increment accordingly.

When time expires (time_sa_mad) for a MAD in the queue, it will be removed from the queue and the user will be notified of the item expiration.

This features is implemented per CA port.

The SA MAD congestion control values are configurable using the following sysfs entries:

```
/sys/class/infiniband/mlx5_0/mad_sa_cc/
    1
      drops
      max_outstanding
      queue_size
```

```
  time_sa_mad
2
drops
max_outstanding
queue_size
time_sa_mad
```

To print the current value:

```
cat /sys/class/infiniband/mlx5_0/mad_sa_cc/1/max_outstanding 16
```

To change the current value:

```
echo 32 > /sys/class/infiniband/mlx5_0/mad_sa_cc/1/max_outstanding
cat      /sys/class/infiniband/mlx5_0/mad_sa_cc/1/max_outstanding
32
```

To reset the drops counter:

```
echo 0 > /sys/class/infiniband/mlx5_0/mad_sa_cc/1/drops
```

Parameters' Valid Ranges

Parameter	Range		Default Values
	MIN	MAX	
max_oustanding	1	2^20	16
queue_size	16	2^20	16
time_sa_mad	1 milliseconds	10000	20 milliseconds

3.3.3 Storage Protocols

There are several storage protocols that use the advantage of InfiniBand and RDMA for performance reasons (high throughput, low latency and low CPU utilization). In this chapter we will discuss the following protocols:

- SCSI RDMA Protocol (SRP) is designed to take full advantage of the protocol off-load and RDMA features provided by the InfiniBand architecture.

- iSCSI Extensions for RDMA (iSER) is an extension of the data transfer model of iSCSI, a storage networking standard for TCP/IP. It uses the iSCSI components while taking the advantage of the RDMA protocol suite. ISER is implemented on various stor- age targets such as TGT, LIO, SCST and out of scope of this manual.
  For various ISER targets configuration steps, troubleshooting and debugging, as well as other implementation of storage protocols over RDMA (such as Ceph over RDMA, nbdX and more) refer to Storage Solutions on the Community website.
- Lustre is an open-source, parallel distributed file system, generally used for large-scale cluster computing that supports many requirements of leadership class HPC simulation environments.
- NVM Express™ over Fabrics (NVME-oF)
  - NVME-oF is a technology specification for networking storage designed to enable NVMe message-based commands to transfer data between a host computer and a target solid-state storage device or system over a network such as Ethernet, Fibre Channel, and InfiniBand. Tunneling NVMe commands through an RDMA fabric provides a high throughput and a low latency. This is an alternative to the SCSi based storage networking protocols.
  - NVME-oF Target Offload is an implementation of the new NVME-oF standard Target (server) side in hardware. Starting from ConnectX-5 family cards, all regular IO requests can be processed by the HCA, with the HCA sending IO requests directly to a real NVMe PCI device, using peer-to-peer PCI communications. This means that excluding connection management and error flows, no CPU utilization will be observed during NVME-oF traffic.
    For further information, please refer to Storage Solutions on the Community website (enterprise-support.nvidia.com/s/).

3.3.3.1 SRP - SCSI RDMA Protocol

The SCSI RDMA Protocol (SRP) is designed to take full advantage of the protocol offload and RDMA features provided by the InfiniBand architecture. SRP allows a large body of SCSI software to be readily used on InfiniBand architecture. The SRP Initiator controls the connection to an SRP Target in order to provide access to remote storage devices across an InfiniBand fabric. The kSRP Target resides in an IO unit and provides storage services.

3.3.3.1.1 SRP Initiator

This SRP Initiator is based on open source from OpenFabrics (www.openfabrics.org) that implements the SCSI RDMA Protocol-2 (SRP-2). SRP-2 is described in Document # T10/1524-D available from http://www.t10.org.

The SRP Initiator supports

- Basic SCSI Primary Commands -3 (SPC-3) (www.t10.org/ftp/t10/drafts/spc3/spc3r21b.pdf)
- Basic SCSI Block Commands -2 (SBC-2) (www.t10.org/ftp/t10/drafts/sbc2/sbc2r16.pdf)
- Basic functionality, task management and limited error handling

⚠ This package, however, does not include an SRP Target.

3.3.3.1.1.1 Loading SRP Initiator

To load the SRP module either:

- Execute the `modprobe ib_srp` command after the OFED driver is up.

or

1. Change the value of `SRP_LOAD` in `/etc/infiniband/openib.conf` to "`yes`".
2. Run `/etc/init.d/openibd restart` for the changes to take effect.

⚠ When loading the `ib_srp` module, it is possible to set the module parameter `srp_sg_tablesize.` This is the maximum number of gather/scatter entries per I/O (default: 12).

3.3.3.1.1.2 SRP Module Parameters

When loading the SRP module, the following parameters can be set (viewable by the "modinfo ib_srp" command):

`cmd_sg_entries`	Default number of gather/scatter entries in the SRP command (default is 12, max 255)
`allow_ext_sg`	Default behavior when there are more than cmd_sg_entries S/G entries after mapping; fails the request when false (default false)
`topspin_workarounds`	Enable workarounds for Topspin/Cisco SRP target bugs
`reconnect_delay`	Time between successive reconnect attempts. Time between successive reconnect attempts of SRP initiator to a disconnected target until dev_loss_tmo timer expires (if enabled), after that the SCSI target will be removed
`fast_io_fail_tmo`	Number of seconds between the observation of a transport layer error and failing all I/O. Increasing this timeout allows more tolerance to transport errors, however, doing so increases the total failover time in case of serious transport failure. Note: fast_io_fail_tmo value must be smaller than the value of recon- nect_delay
`dev_loss_tmo`	Maximum number of seconds that the SRP transport should insulate transport layer errors. After this time has been exceeded the SCSI target is removed. Normally it is advised to set this to -1 (disabled) which will never remove the scsi_host. In deployments where different SRP targets are connected and disconnected frequently, it may be required to enable this timeout in order to clean old scsi_hosts representing targets that no longer exists

Constraints between parameters:

- dev_loss_tmo, fast_io_fail_tmo, reconnect_delay cannot be all disabled or negative values.

- reconnect_delay must be positive number.
- fast_io_fail_tmo must be smaller than SCSI block device timeout.
- fast_io_fail_tmo must be smaller than dev_loss_tmo.

3.3.3.1.1.3 SRP Remote Ports Parameters

Several SRP remote ports parameters are modifiable online on existing connection.

To modify dev_loss_tmo to 600 seconds:

```
echo 600 > /sys/class/srp_remote_ports/port-xxx/dev_loss_tmo
```

To modify fast_io_fail_tmo to 15 seconds:

```
echo 15 > /sys/class/srp_remote_ports/port-xxx/fast_io_fail_tmo
```

To modify reconnect_delay to 10 seconds:

```
echo 20 > /sys/class/srp_remote_ports/port-xxx/reconnect_delay
```

3.3.3.1.1.4 Manually Establishing an SRP Connection

The following steps describe how to manually load an SRP connection between the Initiator and an SRP Target. "Automatic Discovery and Connection to Targets" section explains how to do this automatically.

- Make sure that the ib_srp module is loaded, the SRP Initiator is reachable by the SRP Target, and that an SM is running.
- To establish a connection with an SRP Target and create an SRP (SCSI) device for that target under /dev, use the following command:

```
echo -n id_ext=[GUID value],ioc_guid=[GUID value],dgid=[port GID value],\
pkey=ffff,service_id=[service[0] value] > \
/sys/class/infiniband_srp/srp-mlx[hca number]-[port number]/add_target
```

See "SRP Tools - ibsrpdm, srp_daemon and srpd Service Script" section for instructions on how the parameters in this echo command may be obtained.

Notes:

- Execution of the above "echo" command may take some time
- The SM must be running while the command executes
- It is possible to include additional parameters in the echo command:

- max_cmd_per_lun - Default: 62
- max_sect (short for max_sectors) - sets the request size of a command
- io_class - Default: 0x100 as in rev 16A of the specification (In rev 10 the default was 0xff00)
- tl_retry_count - a number in the range 2..7 specifying the IB RC retry count. Default: 2
- comp_vector, a number in the range 0..n-1 specifying the MSI-X completion vector. Some HCA's allocate multiple (n) MSI-X vectors per HCA port. If the IRQ affinity masks of these interrupts have been configured such that each MSI-X interrupt is handled by a different CPU then the comp_vector parameter can be used to spread the SRP completion workload over multiple CPU's.
- cmd_sg_entries, a number in the range 1..255 that specifies the maximum number of data buffer descriptors stored in the SRP_CMD information unit itself. With allow_ext_sg=0 the parameter cmd_sg_entries defines the maximum S/G list length for a single SRP_CMD, and commands whose S/G list length exceeds this limit after S/G list collapsing will fail.
- initiator_ext - see "Multiple Connections from Initiator InfiniBand Port to the Target" section.

- To list the new SCSI devices that have been added by the echo command, you may use either of the following two methods:
  - Execute “fdisk -l”. This command lists all devices; the new devices are included in this listing.
  - Execute “dmesg” or look at /var/log/messages to find messages with the names of the new devices.

3.3.3.1.1.5 SRP sysfs Parameters

Interface for making ib_srp connect to a new target. One can request ib_srp to connect to a new target by writing a comma-separated list of login parameters to this sysfs attribute. The supported parameters are:

id_ext	A 16-digit hexadecimal number specifying the eight byte identifier extension in the 16-byte SRP target port identifier. The target port identifier is sent by ib_srp to the target in the SRP_LOGIN_REQ request.
ioc_guid	A 16-digit hexadecimal number specifying the eight byte I/O controller GUID portion of the 16-byte target port identifier.
dgid	A 32-digit hexadecimal number specifying the destination GID.
pkey	A four-digit hexadecimal number specifying the InfiniBand partition key.
service_id	A 16-digit hexadecimal number specifying the InfiniBand service ID used to establish communication with the SRP target. How to find out the value of the service ID is specified in the documentation of the SRP target.
max_sect	A decimal number specifying the maximum number of 512-byte sectors to be transferred via a single SCSI command.
max_cmd_per_lun	A decimal number specifying the maximum number of outstanding commands for a single LUN.
io_class	A hexadecimal number specifying the SRP I/O class. Must be either 0xff00 (rev 10) or 0x0100 (rev 16a). The I/O class defines the format of the SRP initiator and target port identifiers.
initiator_ext	A 16-digit hexadecimal number specifying the identifier extension portion of the SRP initiator port identifier. This data is sent by the initiator to the target in the SRP_LOGIN_REQ request.

cmd_sg_entries	A number in the range 1..255 that specifies the maximum number of data buffer descriptors stored in the SRP_CMD information unit itself. With allow_ext_sg=0 the parameter cmd_sg_entries defines the maxi- mum S/G list length for a single SRP_CMD, and commands whose S/G list length exceeds this limit after S/G list collapsing will fail.
allow_ext_sg	Whether ib_srp is allowed to include a partial memory descriptor list in an SRP_CMD instead of the entire list. If a partial memory descriptor list has been included in an SRP_CMD the remaining memory descriptors are communicated from initiator to target via an additional RDMA transfer. Setting allow_ext_sg to 1 increases the maximum amount of data that can be transferred between initiator and target via a single SCSI command. Since not all SRP target implementations support partial memory descriptor lists the default value for this option is 0.
sg_tablesize	A number in the range 1..2048 specifying the maximum S/G list length the SCSI layer is allowed to pass to ib_srp. Specifying a value that exceeds cmd_sg_entries is only safe with partial memory descriptor list support enabled (allow_ext_sg=1).
comp_vector	A number in the range 0..n-1 specifying the MSI-X completion vector. Some HCA's allocate multiple (n) MSI-X vectors per HCA port. If the IRQ affinity masks of these interrupts have been configured such that each MSI-X interrupt is handled by a different CPU then the comp_vector parameter can be used to spread the SRP completion workload over multiple CPU's.
tl_retry_count	A number in the range 2..7 specifying the IB RC retry count.

3.3.3.1.1.6 SRP Tools - ibsrpdm, srp_daemon and srpd Service Script

The OFED distribution provides two utilities: ibsrpdm and srp_daemon:

- They detect targets on the fabric reachable by the Initiator (Step 1)
- Output target attributes in a format suitable for use in the above “echo” command (Step 2)
- A service script srpd which may be started at stack startup

The utilities can be found under /usr/sbin/, and are part of the srptools RPM that may be installed using the OFED installation. Detailed information regarding the various options for these utilities are provided by their man pages.
Below, several usage scenarios for these utilities are presented.

ibsrpdm

ibsrpdm has the following tasks:

1. Detecting reachable targets.
    a. To detect all targets reachable by the SRP initiator via the default umad device (/sys/class/infiniband_mad/umad0), execute the following command:

```
ibsrpdm
```

This command will result into readable output information on each SRP Target detected. Sample:

```
IO Unit Info:
     port LID:        0103
     port GID:        fe80000000000000002c90200402bd5
     change ID:       0002
     max controllers: 0x10
  controller[  1]
     GUID:      0002c90200402bd4
     vendor ID: 0002c9
     device ID: 005a44
     IO class : 0100
     ID:        LSI Storage Systems SRP Driver 200400a0b81146a1
     service entries: 1
```

```
            service[  0]: 200400a0b81146a1 / SRP.T10:200400A0B81146A1
```

b. To detect all the SRP Targets reachable by the SRP Initiator via another umad device, use the following command:

```
ibsrpdm -d <umad device>
```

2. Assisting in SRP connection creation.
   a. To generate an output suitable for utilization in the “echo” command in “Manually Establishing an SRP Connection” section, add the ‘-c’ option to ibsrpdm:

```
ibsrpdm -c
```

Sample output:

```
id_ext=200400A0B81146A1,ioc_guid=0002c90200402bd4,
dgid=fe8000000000000002c90200402bd5,pkey=ffff,service_id=200400a0b81146a1
```

   b. To establish a connection with an SRP Target using the output from the ‘ibsrpdm -c’ example above, execute the following command:

```
echo -n id_ext=200400A0B81146A1,ioc_guid=0002c90200402bd4,
dgid=fe8000000000000002c90200402bd5,pkey=ffff,service_id=200400a0b81146a1 > /sys/
class/infiniband_srp/srp-mlx5_0-1/add_target
```

The SRP connection should now be up; the newly created SCSI devices should appear in the listing obtained from the ‘`fdisk -l`’ command.

3. Discover reachable SRP Targets given an InfiniBand HCA name and port, rather than by just running `/sys/class/infiniband_mad/umad<N>` where `<N>` is a digit.

srpd

The srpd service script allows automatic activation and termination of the srp_daemon utility on all system live InfiniBand ports.

srp_daemon

srp_daemon utility is based on ibsrpdm and extends its functionality. In addition to the ibsrpdm functionality described above, srp_daemon can:

- Establish an SRP connection by itself (without the need to issue the “echo” command described in “Manually Establishing an SRP Connection” section)
- Continue running in background, detecting new targets and establishing SRP connections with them (daemon mode)
- Discover reachable SRP Targets given an infiniband HCA name and port, rather than just by `/dev/umad<N>` where `<N>` is a digit
- Enable High Availability operation (together with Device-Mapper Multipath)
- Have a configuration file that determines the targets to connect to:

1. srp_daemon commands equivalent to ibsrpdm:

```
"srp_daemon -a -o" is equivalent to "ibsrpdm"
"srp_daemon -c -a -o" is equivalent to "ibsrpdm -c"
```

Note: These srp_daemon commands can behave differently than the equivalent ibsrpdm command when /etc/srp_daemon.conf is not empty.

2. srp_daemon extensions to ibsrpdm.

    - To discover SRP Targets reachable from the HCA device <InfiniBand HCA name> and the port <port num>, (and to generate output suitable for 'echo'), execute:

      ```
      host1# srp_daemon -c -a -o -i <InfiniBand HCA name> -p <port number>
      ```

      Note: To obtain the list of InfiniBand HCA device names, you can either use the ibstat tool or run ‘ls /sys/class/infiniband’.
    - To both discover the SRP Targets and establish connections with them, just add the -e option to the above command.
    - Executing srp_daemon over a port without the -a option will only display the reachable targets via the port and to which the initiator is not connected. If executing with the -e option it is better to omit -a.
    - It is recommended to use the -n option. This option adds the initiator_ext to the connecting string (see "Multiple Connections from Initiator InfiniBand Port to the Target" section).
    - srp_daemon has a configuration file that can be set, where the default is /etc/srp_daemon.conf. Use the -f to supply a different configuration file that configures the targets srp_daemon is allowed to connect to. The configuration file can also be used to set values for additional parameters (e.g., max_cmd_per_lun, max_sect).
    - A continuous background (daemon) operation, providing an automatic ongoing detection and connection capability. See "Automatic Discovery and Connection to Targets" section.

3.3.3.1.1.7 Automatic Discovery and Connection to Targets

- Make sure the ib_srp module is loaded, the SRP Initiator can reach an SRP Target, and that an SM is running.
- To connect to all the existing Targets in the fabric, run “`srp_daemon -e -o`”. This utility will scan the fabric once, connect to every Target it detects, and then exit.

  > ⚠ srp_daemon will follow the configuration it finds in /etc/srp_daemon.conf. Thus, it will ignore a target that is disallowed in the configuration file.

- To connect to all the existing Targets in the fabric and to connect to new targets that will join the fabric, execute srp_daemon -e. This utility continues to execute until it is either killed by the user or encounters connection errors (such as no SM in the fabric).
- To execute SRP daemon as a daemon on all the ports:
    - srp_daemon.sh (found under /usr/sbin/). srp_daemon.sh sends its log to /var/log/srp_daemon.log.
    - Start the srpd service script, run service srpd start

For the changes in openib.conf to take effect, run:

```
/etc/init.d/openibd restart
```

3.3.3.1.1.8 Multiple Connections from Initiator InfiniBand Port to the Target

Some system configurations may need multiple SRP connections from the SRP Initiator to the same SRP Target: to the same Target IB port, or to different IB ports on the same Target HCA.
In case of a single Target IB port, i.e., SRP connections use the same path, the configuration is enabled using a different initiator_ext value for each SRP connection. The initiator_ext value is a 16-hexadecimal-digit value specified in the connection command.

Also in case of two physical connections (i.e., network paths) from a single initiator IB port to two different IB ports on the same Target HCA, there is need for a different initiator_ext value on each path. The conventions is to use the Target port GUID as the initiator_ext value for the relevant path.

If you use srp_daemon with -n flag, it automatically assigns initiator_ext values according to this convention. For example:

```
id_ext=200500A0B81146A1,ioc_guid=0002c90200402bec,\
dgid=fe8000000000000002c90200402bed,pkey=ffff,\ service_id=200500a0b81146a1,initiator_ext=ed2b400002c90200
```

Notes:

- It is recommended to use the -n flag for all srp_daemon invocations.
- ibsrpdm does not have a corresponding option.
- srp_daemon.sh always uses the -n option (whether invoked manually by the user, or automatically at startup by setting SRP_DAEMON_ENABLE to yes).

3.3.3.1.1.9 High Availability (HA)

High Availability works using the Device-Mapper (DM) multipath and the SRP daemon. Each initiator is connected to the same target from several ports/HCAs. The DM multipath is responsible for joining together different paths to the same target and for failover between paths when one of them goes offline. Multipath will be executed on newly joined SCSI devices.
Each initiator should execute several instances of the SRP daemon, one for each port. At startup, each SRP daemon detects the SRP Targets in the fabric and sends requests to the ib_srp module to connect to each of them. These SRP daemons also detect targets that subsequently join the fabric, and send the ib_srp module requests to connect to them as well.

Operation

When a path (from port1) to a target fails, the ib_srp module starts an error recovery process. If this process gets to the reset_host stage and there is no path to the target from this port, ib_srp will remove this scsi_host. After the scsi_host is removed, multipath switches to another path to this target (from another port/HCA).
When the failed path recovers, it will be detected by the SRP daemon. The SRP daemon will then request ib_srp to connect to this target. Once the connection is up, there will be a new scsi_host for this target. Multipath will be executed on the devices of this host, returning to the original state (prior to the failed path).

Manual Activation of High Availability

Initialization - execute after each boot of the driver:

1. Execute `modprobe dm-multipath`
2. Execute `modprobe ib-srp`

3. Make sure you have created file /etc/udev/rules.d/91-srp.rules as described above
4. Execute for each port and each HCA:

```
srp_daemon -c -e -R 300 -i <InfiniBand HCA name> -p <port number>
```

This step can be performed by executing srp_daemon.sh, which sends its log to /var/log/srp_daemon.log.

Now it is possible to access the SRP LUNs on /dev/mapper/.

⚠ It is possible for regular (non-SRP) LUNs to also be present; the SRP LUNs may be identified by their names. You can configure the /etc/multipath.conf file to change multipath behavior.

⚠ It is also possible that the SRP LUNs will not appear under /dev/mapper/. This can occur if the SRP LUNs are in the black-list of multipath. Edit the 'blacklist' section in /etc/multipath.conf and make sure the SRP LUNs are not blacklisted.

Automatic Activation of High Availability

- Start srpd service, run:

```
service srpd start
```

- From the next loading of the driver it will be possible to access the SRP LUNs on /dev/mapper/

  ⚠ It is possible that regular (not SRP) LUNs are also present. SRP LUNs may be identified by their name.

- It is possible to see the output of the SRP daemon in /var/log/srp_daemon.log

3.3.3.1.2 Shutting Down SRP

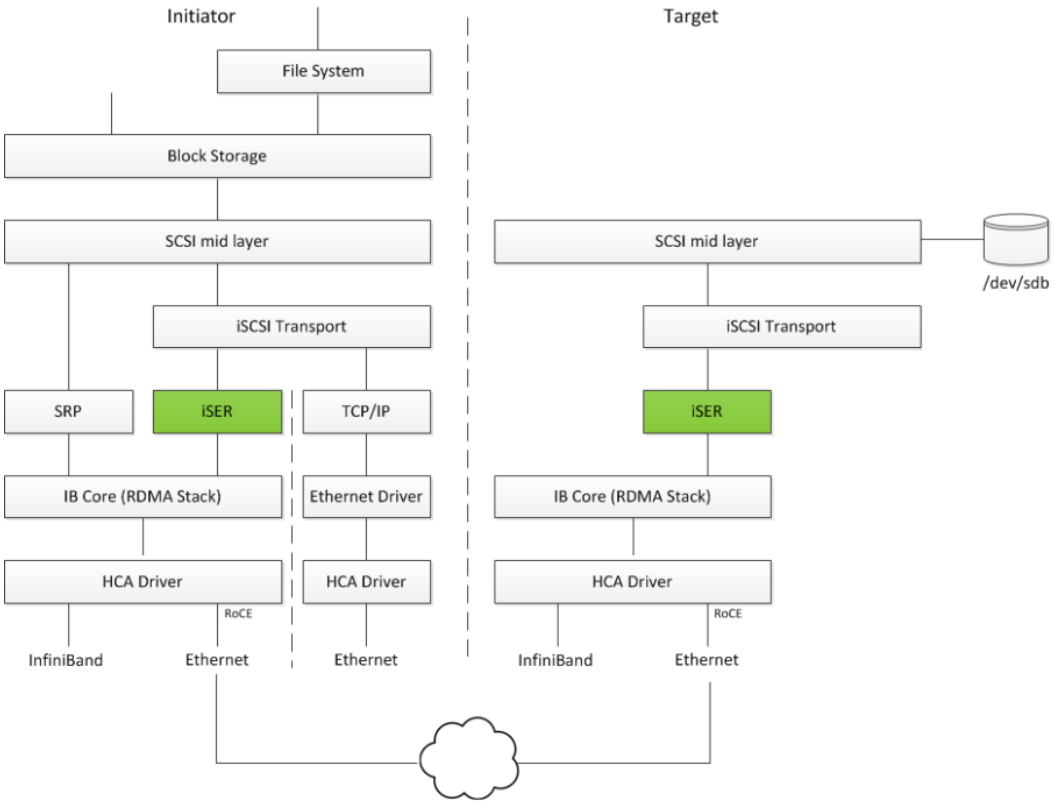
SRP can be shutdown by using "rmmod ib_srp", or by stopping the OFED driver ("/etc/init.d/openibd stop"), or as a by-product of a complete system shutdown.
Prior to shutting down SRP, remove all references to it. The actions you need to take depend on the way SRP was loaded. There are three cases:

1. Without High Availability
   When working without High Availability, you should unmount the SRP partitions that were mounted prior to shutting down SRP.
2. After Manual Activation of High Availability
   If you manually activated SRP High Availability, perform the following steps:
   a. Unmount all SRP partitions that were mounted.
   b. Stop service srpd (Kill the SRP daemon instances).
   c. Make sure there are no multipath instances running. If there are multiple instances, wait for them to end or kill them.
   d. Run: `multipath -F`

3. After Automatic Activation of High Availability
   If SRP High Availability was automatically activated, SRP shutdown must be part of the driver shutdown ("/etc/init.d/openibd stop") which performs Steps 2-4 of case b above. However, you still have to unmount all SRP partitions that were mounted before driver shutdown.

3.3.3.2 iSCSI Extensions for RDMA (iSER)

iSCSI Extensions for RDMA (iSER) extends the iSCSI protocol to RDMA. It permits data to be transferred directly into and out of SCSI buffers without intermediate data copies.
iSER uses the RDMA protocol suite to supply higher bandwidth for block storage transfers (zero time copy behavior). To that fact, it eliminates the TCP/IP processing overhead while preserving the compatibility with iSCSI protocol.



There are three target implementation of ISER:

- Linux SCSI target framework (tgt)
- Linux-IO target (LIO)
- Generic SCSI target subsystem for Linux (SCST)

Each one of those targets can work in TCP or iSER transport modes.
iSER also supports RoCE without any additional configuration required. To bond the RoCE interfaces, set the fail_over_mac option in the bonding driver (see "Bonding IPoIB").
RDMA/RoCE is located below the iSER block on the network stack. In order to run iSER, the RDMA layer should be configured and validated (over Ethernet or InfiniBand). For troubleshooting RDMA, please refer to "HowTo Enable, Verify and Troubleshoot RDMA" on the Community website.

3.3.3.2.1 iSER Initiator

The iSER initiator is controlled through the iSCSI interface available from the iscsi-initiator-utils package.

To discover and log into iSCSI targets, as well as access and manage the open-iscsi database use the `iscasiadm` utility, a command-line tool.

To enable iSER as a transport protocol use "`I iser`" as a parameter of the `iscasiadm` command.

Example for discovering and connecting targets over iSER:

```
iscsiadm -m discovery -o new -o old -t st -I iser -p <ip:port> -l
```

Note that the target implementation (e.g. LIO, SCST, TGT) does not affect he initiator process and configuration.

3.3.3.2.2 iSER Targets

> ⚠ Setting the iSER target is out of scope of this manual. For guidelines of how to do so, please refer to the relevant target documentation (e.g. stgt, targetcli).

Targets settings such as timeouts and retries are set the same as any other iSCSI targets.

> ⚠ If targets are set to auto connect on boot, and targets are unreachable, it may take a long time to continue the boot process if timeouts and max retries are set too high.

For various configuration, troubleshooting and debugging examples, refer to Storage Solutions on the Community website.

3.3.3.3 Lustre

Lustre is an open-source, parallel distributed file system, generally used for large-scale cluster computing that supports many requirements of leadership class HPC simulation environments.

Lustre Compilation for MLNX_OFED:

> ⚠ This procedure applies to RHEL/SLES OSs supported by Lustre. For further information, please refer to Lustre Release Notes.

To compile Lustre version 2.4.0 and higher:

```
$ ./configure --with-o2ib=/usr/src/ofa_kernel/default/
```

```
$ make rpms
```

To compile older Lustre versions:

```
$ EXTRA_LNET_INCLUDE="-I/usr/src/ofa_kernel/default/include/ -include /usr/src/ofa_kernel/default/include/linux/
compat-2.6.h" ./configure --with-o2ib=/usr/src/ofa_kernel/default/
$ EXTRA_LNET_INCLUDE="-I/usr/src/ofa_kernel/default/include/ -include /usr/src/ofa_kernel/default/include/linux/
compat-2.6.h" make rpms
```

For full installation example, refer to HowTo Install NVIDIA OFED driver for Lustre Community post.

3.3.3.4 NVME-oF - NVM Express over Fabrics

3.3.3.4.1 NVME-oF

NVME-oF enables NVMe message-based commands to transfer data between a host computer and a target solid-state storage device or system over a network such as Ethernet, Fibre Channel, and InfiniBand. Tunneling NVMe commands through an RDMA fabric provides a high throughput and a low latency.
For information on how to configure NVME-oF, please refer to the HowTo Configure NVMe over Fabrics Community post.

The --with-nvmf installation option should not be specified, if nvme-tcp kernel module is used. In this case, the native Inbox nvme-tcp kernel module will be loaded.

3.3.3.4.2 NVME-oF Target Offload

This feature is only supported for ConnectX-5 adapter cards family and above.

NVME-oF Target Offload is an implementation of the new NVME-oF standard Target (server) side in hardware. Starting from ConnectX-5 family cards, all regular IO requests can be processed by the HCA, with the HCA sending IO requests directly to a real NVMe PCI device, using peer-to-peer PCI communications. This means that excluding connection management and error flows, no CPU utilization will be observed during NVME-oF traffic.

- For instructions on how to configure NVME-oF target offload, refer to HowTo Configure NVME-oF Target Offload Community post.
- For instructions on how to verify that NVME-oF target offload is working properly, refer to Simple NVMe-oF Target Offload Benchmark Community post.

3.3.4 Virtualization

The chapter contains the following sections:

- Single Root IO Virtualization (SR-IOV)

- Enabling Paravirtualization
- VXLAN Hardware Stateless Offloads
- Q-in-Q Encapsulation per VF in Linux (VST)
- 802.1Q Double-Tagging
- Scalable Functions

3.3.4.1 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. NVIDIA adapters are capable of exposing up to 127 virtual instances (Virtual Functions (VFs) for each port in the NVIDIA ConnectX® family cards. These virtual functions can then be provisioned separately. Each VF can be seen as an additional device connected to the Physical Function. It shares the same resources with the Physical Function, and its number of ports equals those of 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 Red Hat Linux environment using ConnectX® VPI adapter cards.

3.3.4.1.1 System Requirements

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

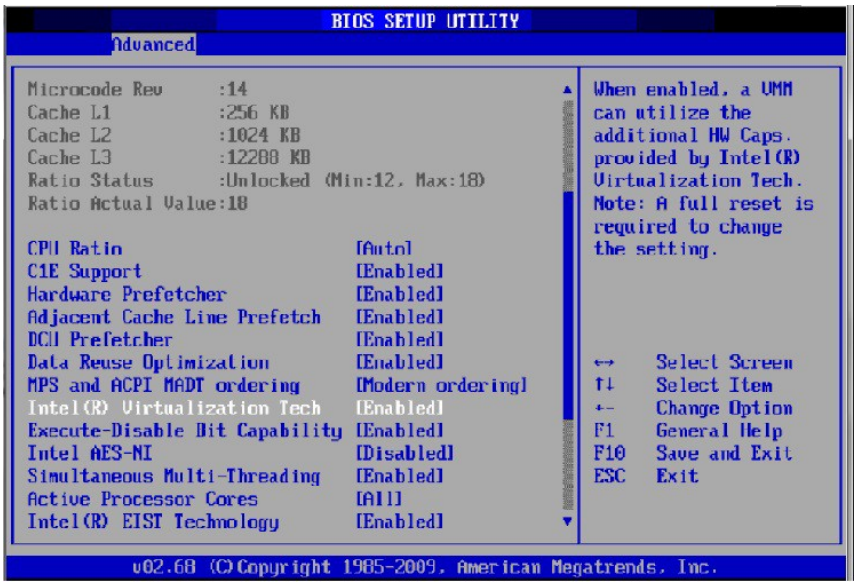
- MLNX_OFED Driver
- A server/blade with an SR-IOV-capable motherboard BIOS
- Hypervisor that supports SR-IOV such as: Red Hat Enterprise Linux Server Version 6
- NVIDIA ConnectX® VPI Adapter Card family with SR-IOV capability

3.3.4.1.2 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:

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

2. Enable "Intel Virtualization Technology".



3. Install a hypervisor that supports SR-IOV.
4. Depending on your system, update the /boot/grub/grub.conf file to include a similar command line load parameter for the Linux kernel.
   For example, to Intel systems, add:

```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (4.x.x)
        root (hd0,0)
        kernel /vmlinuz-4.x.x ro root=/dev/VolGroup00/LogVol00 rhgb quiet
        intel_iommu=on        initrd /initrd-4.x.x.img
```

   Note: Please make sure the parameter "`intel_iommu=on`" exists when updating the /boot/grub/grub.conf file, otherwise SR-IOV cannot be loaded.

Some OSs use /boot/grub2/grub.cfg file. If your server uses such file, please edit this file instead (add “`intel_iommu=on`” for the relevant menu entry at the end of the line that starts with "linux16").

3.3.4.1.3 Configuring SR-IOV (Ethernet)

To set SR-IOV in Ethernet mode, refer to HowTo Configure SR-IOV for ConnectX-4/ConnectX- 5/ ConnectX-6 with KVM (Ethernet) Community Post.

3.3.4.1.4 Configuring SR-IOV (InfiniBand)

1. Install the MLNX_OFED driver for Linux that supports SR-IOV.
2. Check if SR-IOV is enabled in the firmware.

```
mlxconfig -d /dev/mst/mt4115_pciconf0 q

  Device #1:
  ----------

  Device type:    Connect4
  PCI device:     /dev/mst/mt4115_pciconf0
  Configurations:          Current
     SRIOV_EN              1
     NUM_OF_VFS            8
```

> If needed, use mlxconfig to set the relevant fields:
> `mlxconfig -d /dev/mst/mt4115_pciconf0 set SRIOV_EN=1 NUM_OF_VFS=16`

3. Reboot the server.
4. Write to the sysfs file the number of Virtual Functions you need to create for the PF. You can use one of the following equivalent files:
   You can use one of the following equivalent files:
   - A standard Linux kernel generated file that is available in the new kernels.

```
echo [num_vfs] > /sys/class/infiniband/mlx5_0/device/sriov_numvfs
```

   Note: This file will be generated only if IOMMU is set in the grub.conf file (by adding intel_iommu=on, as seen in the fourth step under “Setting Up SR-IOV”).
   - A file generated by the mlx5_core driver with the same functionality as the kernel generated one.

```
echo [num_vfs] > /sys/class/infiniband/mlx5_0/device/mlx5_num_vfs
```

   Note: This file is used by old kernels that do not support the standard file. In such kernels, using sriov_numvfs results in the following error: “bash: echo: write error: Function not implemented”.
   The following rules apply when writing to these files:
   - If there are no VFs assigned, the number of VFs can be changed to any valid value (0 - max #VFs as set during FW burning)
   - If there are VFs assigned to a VM, it is not possible to change the number of VFs
   - If the administrator unloads the driver on the PF while there are no VFs assigned, the driver will unload and SRI-OV will be disabled
   - If there are VFs assigned while the driver of the PF is unloaded, SR-IOV will not be disabled.

This means that VFs will be visible on the VM. However, they will not be operational. This is applicable to OSs with kernels that use pci_stub and not vfio.

- The VF driver will discover this situation and will close its resources
- When the driver on the PF is reloaded, the VF becomes operational. The administrator of the VF will need to restart the driver in order to resume working with the VF.

5. Load the driver. To verify that the VFs were created. Run:

```
lspci | grep Mellanox
08:00.0 Infiniband controller: Mellanox Technologies MT27700 Family [ConnectX-4]
08:00.1 Infiniband controller: Mellanox Technologies MT27700 Family [ConnectX-4]
08:00.2 Infiniband controller: Mellanox Technologies MT27700 Family [ConnectX-4 Virtual Function]
08:00.3 Infiniband controller: Mellanox Technologies MT27700 Family [ConnectX-4 Virtual Function]
08:00.4 Infiniband controller: Mellanox Technologies MT27700 Family [ConnectX-4 Virtual Function]
08:00.5 Infiniband controller: Mellanox Technologies MT27700 Family [ConnectX-4 Virtual Function]
```

6. Configure the VFs.
   After VFs are created, 3 sysfs entries per VF are available under /sys/class/infiniband/mlx5_<PF INDEX>/device/sriov (shown below for VFs 0 to 2):

```
+-- 0
|   +-- node
|   +-- policy
|   +-- port
+-- 1
|   +-- node
|   +-- policy
|   +-- port
+-- 2
    +-- node
    +-- policy
    +-- port
```

   For each Virtual Function, the following files are available:
   - Node - Node's GUID:
   The user can set the node GUID by writing to the /sys/class/infiniband/<PF>/device/sriov/<index>/node file. The example below, shows how to set the node GUID for VF 0 of mlx5_0.

```
echo 00:11:22:33:44:55:1:0 > /sys/class/infiniband/mlx5_0/device/sriov/0/node
```

   - Port - Port's GUID:
   The user can set the port GUID by writing to the /sys/class/infiniband/<PF>/device/sriov/<index>/port file. The example below, shows how to set the port GUID for VF 0 of mlx5_0.

```
echo 00:11:22:33:44:55:2:0 > /sys/class/infiniband/mlx5_0/device/sriov/0/port
```

   - Policy - The vport's policy. The user can set the port GUID by writing to the /sys/class/infiniband/<PF>/device/sriov/<index>/port file. The policy can be one of:
   - Down - the VPort PortState remains 'Down'
   - Up - if the current VPort PortState is 'Down', it is modified to 'Initialize'. In all other states, it is unmodified. The result is that the SM may bring the VPort up.
   - Follow - follows the PortState of the physical port. If the PortState of the physical port is 'Active', then the VPort implements the 'Up' policy. Otherwise, the VPort PortState is 'Down'.
   Notes:
   - The policy of all the vports is initialized to "Down" after the PF driver is restarted except for VPort0 for which the policy is modified to 'Follow' by the PF driver.
   - To see the VFs configuration, you must unbind and bind them or reboot the VMs if the VFs were assigned.

7. Make sure that OpenSM supports Virtualization (Virtualization must be enabled).
   The /etc/opensm/opensm.conf file should contain the following line:

```
virt_enabled 2
```

Note: OpenSM and any other utility that uses SMP MADs (ibnetdiscover, sminfo, iblink- info, smpdump, ibqueryerr, ibdiagnet and smpquery) should run on the PF and not on the VFs. In case of multi PFs (multi-host), OpenSM should run on Host0.

3.3.4.1.4.1 VFs Initialization Note

Since the same mlx5_core driver supports both Physical and Virtual Functions, once the Virtual Functions are created, the driver of the PF will attempt to initialize them so they will be available to the OS owning the PF. If you want to assign a Virtual Function to a VM, you need to make sure the VF is not used by the PF driver. If a VF is used, you should first unbind it before assigning to a VM.

To unbind a device use the following command:

1. Get the full PCI address of the device.

```
lspci -D
```

Example:

```
0000:09:00.2
```

2. Unbind the device.

```
echo 0000:09:00.2 > /sys/bus/pci/drivers/mlx5_core/unbind
```

3. Bind the unbound VF.

```
echo 0000:09:00.2 > /sys/bus/pci/drivers/mlx5_core/bind
```

3.3.4.1.4.2 PCI BDF Mapping of PFs and VFs

PCI addresses are sequential for both of the PF and their VFs. Assuming the card's PCI slot is 05:00 and it has 2 ports, the PFs PCI address will be 05:00.0 and 05:00.1.
Given 3 VFs per PF, the VFs PCI addresses will be:

```
05:00.2-4 for VFs 0-2 of PF 0 (mlx5_0)
05:00.5-7 for VFs 0-2 of PF 1 (mlx5_1)
```

3.3.4.1.5 Additional SR-IOV Configurations

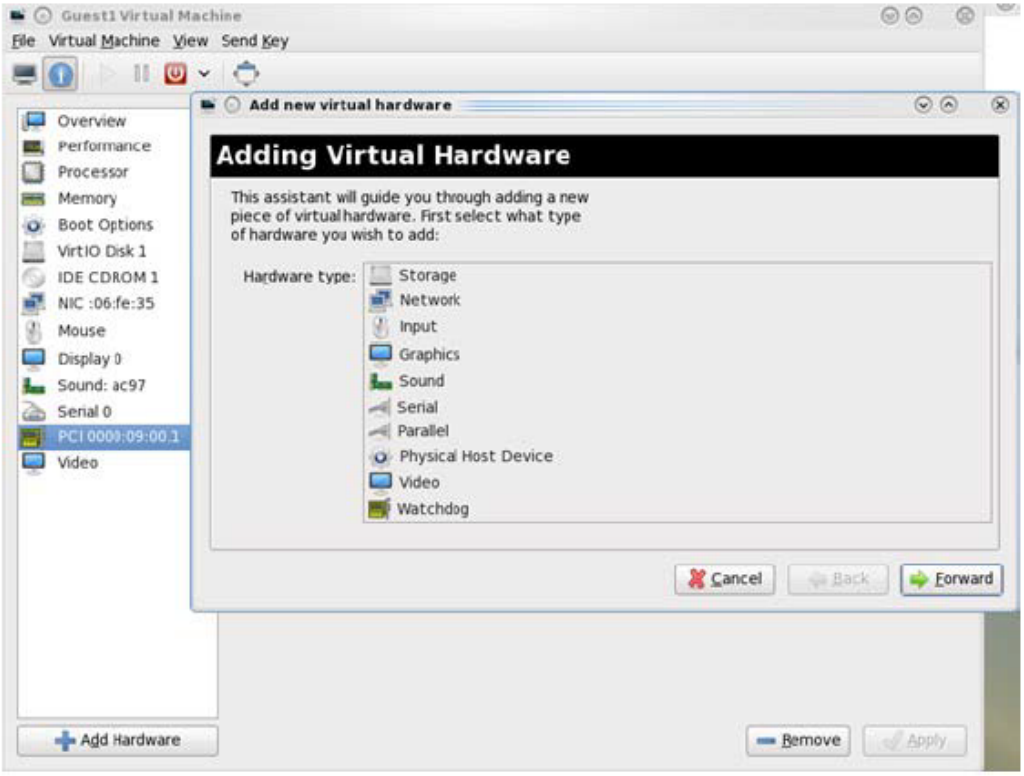
3.3.4.1.5.1 Assigning a Virtual Function to a Virtual Machine

This section describes a mechanism for adding a SR-IOV VF to a Virtual Machine.

3.3.4.1.5.2 Assigning the SR-IOV Virtual Function to the Red Hat KVM VM Server

1. Run the virt-manager.

2. Double click on the virtual machine and open its Properties.
3. Go to Details → Add hardware → PCI host device.



4. Choose a NVIDIA virtual function according to its PCI device (e.g., 00:03.1)
5. If the Virtual Machine is up reboot it, otherwise start it.
6. Log into the virtual machine and verify that it recognizes the NVIDIA card. Run:

```
lspci | grep Mellanox
```

Example:

```
lspci | grep Mellanox
01:00.0 Infiniband controller: Mellanox Technologies MT28800 Family [ConnectX-5 Ex]
```

7. Add the device to the `/etc/sysconfig/network-scripts/ifcfg-ethX` configuration file. The MAC address for every virtual function is configured randomly, therefore it is not necessary to add it.

3.3.4.1.5.3 Ethernet Virtual Function Configuration when Running SR-IOV

SR-IOV Virtual function configuration can be done through Hypervisor iprout2/netlink tool, if present. Otherwise, it can be done via sysfs.

```
ip link set { dev DEVICE | group DEVGROUP } [ { up | down } ]
...
[ vf NUM [ mac LLADDR ] [ vlan VLANID [ qos VLAN-QOS ] ]
...
[ spoofchk { on | off} ] ]
...

sysfs configuration (ConnectX-4):
```

```
/sys/class/net/enp8s0f0/device/sriov/[VF]

+-- [VF]
| +-- config
| +-- link_state
| +-- mac
| +-- mac_list
| +-- max_tx_rate
| +-- min_tx_rate
| +-- spoofcheck
| +-- stats
| +-- trunk
| +-- trust
| +-- vlan
```

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.
The default behavior is VGT.

To configure VF VST mode, run:

```
ip link set dev <PF device> vf <NUM> vlan <vlan_id> [qos <qos>]
```

where:

- NUM = 0..max-vf-num
- vlan_id = 0..4095
- qos = 0..7

For example:

- ip link set dev eth2 vf 2 vlan 10 qos 3 - sets VST mode for VF #2 belonging to PF eth2, with vlan_id = 10 and qos = 3
- ip link set dev eth2 vf 2 vlan 0 - sets mode for VF 2 back to VGT

Additional Ethernet VF Configuration Options

- Guest MAC configuration - by default, guest MAC addresses are configured to be all zeroes. If the administrator wishes the guest to always start up with the same MAC, he/she should configure guest MACs before the guest driver comes up. The guest MAC may be configured by using:

  ```
  ip link set dev <PF device> vf <NUM> mac <LLADDR>
  ```

  For legacy and ConnectX-4 guests, which do not generate random MACs, the administrator should always configure their MAC addresses via IP link, as above.

- Spoof checking - Spoof checking is currently available only on upstream kernels newer than 3.1.

  ```
  ip link set dev <PF device> vf <NUM> spoofchk [on | off]
  ```

- Guest Link State

```
ip link set dev <PF device> vf <UM> state [enable| disable| auto]
```

Virtual Function Statistics

Virtual function statistics can be queried via sysfs:

```
cat /sys/class/infiniband/mlx5_2/device/sriov/2/stats tx_packets : 5011
tx_bytes : 4450870
tx_dropped : 0
rx_packets : 5003
rx_bytes : 4450222
rx_broadcast : 0
rx_multicast : 0
tx_broadcast : 0
tx_multicast : 8
rx_dropped : 0
```

Mapping VFs to Ports

To view the VFs mapping to ports:

Use the ip link tool v2.6.34~3 and above.

```
ip link
```

Output:

```
61: p1p1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 00:02:c9:f1:72:e0 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
    vf 37 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
    vf 38 MAC ff:ff:ff:ff:ff:ff, vlan 65535, spoof checking off, link-state disable
    vf 39 MAC ff:ff:ff:ff:ff:ff, vlan 65535, spoof checking off, link-state disable
```

When a MAC is ff:ff:ff:ff:ff:ff, the VF is not assigned to the port of the net device it is listed under. In the example above, vf38 is not assigned to the same port as p1p1, in contrast to vf0. However, even VFs that are not assigned to the net device, could be used to set and change its settings. For example, the following is a valid command to change the spoof check:

```
ip link set dev p1p1 vf 38 spoofchk on
```

This command will affect only the vf38. The changes can be seen in ip link on the net device that this device is assigned to.

RoCE Support

RoCE is supported on Virtual Functions and VLANs may be used with it. For RoCE, the hypervisor GID table size is of 16 entries while the VFs share the remaining 112 entries. When the number of VFs is larger than 56 entries, some of them will have GID table with only a single entry which is inadequate if VF's Ethernet device is assigned with an IP address.

3.3.4.1.5.4 Virtual Guest Tagging (VGT+)

VGT+ is an advanced mode of Virtual Guest Tagging (VGT), in which a VF is allowed to tag its own packets as in VGT, but is still subject to an administrative VLAN trunk policy. The policy determines which VLAN IDs are allowed to be transmitted or received. The policy does not determine the user priority, which is left unchanged.
Packets can be sent in one of the following modes: when the VF is allowed to send/receive untagged and priority tagged traffic and when it is not. No default VLAN is defined for VGT+ port. The send packets are passed to the eSwitch only if they match the set, and the received packets are forwarded to the VF only if they match the set.

Configuration

When working in SR-IOV, the default operating mode is VGT.

To enable VGT+ mode:

Set the corresponding port/VF (in the example below port eth5, VF0) range of allowed VLANs.

```
echo "<add> <start_vid> <end_vid>" > /sys/class/net/eth5/device/sriov/0/trunk
```

Examples:

- Adding VLAN ID range (4-15) to trunk:

```
echo add 4 15 > /sys/class/net/eth5/device/sriov/0/trunk
```

- Adding a single VLAN ID to trunk:

```
echo add 17 17 > /sys/class/net/eth5/device/sriov/0/trunk
```

Note: When VLAN ID = 0, it indicates that untagged and priority-tagged traffics are allowed

To disable VGT+ mode, make sure to remove all VLANs.

```
echo rem 0 4095 > /sys/class/net/eth5/device/sriov/0/trunk
```

To remove selected VLANs.

- Remove VLAN ID range (4-15) from trunk:

```
echo rem 4 15 > /sys/class/net/eth5/device/sriov/0/trunk
```

- Remove a single VLAN ID from trunk:

```
echo rem 17 17 > /sys/class/net/eth5/device/sriov/0/trunk
```

3.3.4.1.5.5 SR-IOV Advanced Security Features

SR-IOV MAC Anti-Spoofing

Normally, MAC addresses are unique identifiers assigned to network interfaces, and they are fixed addresses that cannot be changed. MAC address spoofing is a technique for altering the MAC address to serve different purposes. Some of the cases in which a MAC address is altered can be legal, while others can be illegal and abuse security mechanisms or disguises a possible attacker.
The SR-IOV MAC address anti-spoofing feature, also known as MAC Spoof Check provides protection against malicious VM MAC address forging. If the network administrator assigns a MAC address to a VF (through the hypervisor) and enables spoof check on it, this will limit the end user to send traffic only from the assigned MAC address of that VF.

MAC Anti-Spoofing Configuration

⚠ MAC anti-spoofing is disabled by default.

In the configuration example below, the VM is located on VF-0 and has the following MAC address: 11:22:33:44:55:66.
There are two ways to enable or disable MAC anti-spoofing:

1. Use the standard IP link commands - available from Kernel 3.10 and above.
   a. To enable MAC anti-spoofing, run:

   ```
   ip link set ens785f1 vf 0 spoofchk on
   ```

   b. To disable MAC anti-spoofing, run:

   ```
   ip link set ens785f1 vf 0 spoofchk off
   ```

2. Specify echo "ON" or "OFF" to the file located under /sys/class/net/<ifname / device/sriov/<VF index>/spoofcheck.
   a. To enable MAC anti-spoofing, run:

   ```
   echo "ON" > /sys/class/net/ens785f1/vf/0/spoofchk
   ```

   b. To disable MAC anti-spoofing, run:

   ```
   echo "OFF" > /sys/class/net/ens785f1/vf/0/spoofchk
   ```

⚠ This configuration is non-persistent and does not survive driver restart.

Limit and Bandwidth Share Per VF

This feature enables rate limiting traffic per VF in SR-IOV mode. For details on how to configure rate limit per VF for ConnectX-4 and above adapter cards, please refer to HowTo Configure Rate Limit per VF for ConnectX-4/ConnectX-5/ConnectX-6 Community post.

Limit Bandwidth per Group of VFs

VFs Rate Limit for vSwitch (OVS) feature allows users to join available VFs into groups and set a rate limitation on each group. Rate limitation on a VF group ensures that the total Tx bandwidth that the VFs in this group get (altogether combined) will not exceed the given value.
With this feature, a VF can still be configured with an individual rate limit as in the past (under /sys/class/net/<ifname>/device/sriov/<vf_num>/max_tx_rate). However, the actual bandwidth limit on the VF will eventually be determined considering the VF group limitation and how many VFs are in the same group.
For example: 2 VFs (0 and 1) are attached to group 3.

Case 1: The rate limitation on the group is set to 20G. Rate limit of each VF is 15G
Result: Each VF will have a rate limit of 10G

Case 2: Group's max rate limitation is still set to 20G. VF 0 is configured to 30G limit, while VF 1 is configured to 5G rate limit
Result: VF 0 will have 15G de-facto. VF 1 will have 5G

The rule of thumb is that the group's bandwidth is distributed evenly between the number of VFs in the group. If there are leftovers, they will be assigned to VFs whose individual rate limit has not been met yet.

VFs Rate Limit Feature Configuration

1. When VF rate group is supported by FW, the driver will create a new hierarchy in the SRI-OV sysfs named "groups" (/sys/class/net/<ifname>/device/sriov/groups/). It will contain all the info and the configurations allowed for VF groups.
2. All VFs are placed in group 0 by default since it is the only existing group following the initial driver start. It would be the only group available under /sys/class/net/<ifname>/device/sriov/groups/
3. The VF can be moved to a different group by writing to the group file -> echo $GROUP_ID > /sys/class/net/<ifname>/device/sriov/<vf_id>/group
4. The group IDs allowed are 0-255
5. Only when there is at least 1 VF in a group, there will be a group configuration available under /sys/class/net/<ifname>/device/sriov/groups/ (Except for group 0, which is always available even when it's empty).
6. Once the group is created (by moving at least 1 VF to that group), users can configure the group's rate limit. For example:
    a. echo 10000 > /sys/class/net/<ifname>/device/sriov/5/max_tx_rate – setting individual rate limitation of VF 5 to 10G (Optional)
    b. echo 7 > /sys/class/net/<ifname>/device/sriov/5/group – moving VF 5 to group 7
    c. echo 5000 > /sys/class/net/<ifname>/device/sriov/groups/7/max_tx_rate – setting group 7 with rate limitation of 5G
    d. When running traffic via VF 5 now, it will be limited to 5G because of the group rate limit even though the VF itself is limited to 10G
    e. echo 3 > /sys/class/net/<ifname>/device/sriov/5/group – moving VF 5 to group 3

f. Group 7 will now disappear from /sys/class/net/<ifname>/device/sriov/groups since there are 0 VFs in it. Group 3 will now appear. Since there's no rate limit on group 3, VF 5 can transmit at 10G (thanks to its individual configuration)

Notes:

- You can see to which group the VF belongs to in the 'stats' sysfs (cat /sys/class/net/<ifname>/device/sriov/<vf_num>/stats)
- You can see the current rate limit and number of attached VFs to a group in the group's 'config' sysfs (cat /sys/class/net/<ifname>/device/sriov/groups/<group_id>/config)

Bandwidth Guarantee per Group of VFs

Bandwidth guarantee (minimum BW) can be set on a group of VFs to ensure this group is able to transmit at least the amount of bandwidth specified on the wire.

Note the following:

- The minimum BW settings on VF groups determine how the groups share the total BW between themselves. It does not impact an individual VF's rate settings.
- The total minimum BW that is set on the VF groups should not exceed the total line rate. Otherwise, results are unexpected.
- It is still possible to set minimum BW on the individual VFs inside the group. This will determine how the VFs share the group's minimum BW between themselves. The total minimum BW of the VF member should not exceed the minimum BW of the group.

For instruction on how to create groups of VFs, see Limit Bandwidth per Group of VFs above.

Example

With a 40Gb link speed, assuming 4 groups and default group 0 have been created:

```
echo 20000 > /sys/class/net/<ifname>/device/sriov/group/1/min_tx_rate
echo 5000 > /sys/class/net/<ifname>/device/sriov/group/2/min_tx_rate
echo 15000 > /sys/class/net/<ifname>/device/sriov/group/3/min_tx_rate
```

```
Group 0(default) : 0 - No BW guarantee is configured.
Group 1 : 20000 - This is the maximum min rate among groups
Group 2 : 5000 which is 25% of the maximum min rate
Group 3 : 15000 which is 75% of the maximum min rate
Group 4 : 0 - No BW guarantee is configured.
```

Assuming there are VFs attempting to transmit in full line rate in all groups, the results would look like: In which case, the minimum BW allocation would be:

```
Group0 - Will have no BW to use since no BW guarantee was set on it while other groups do have such settings.
Group1 - Will transmit at 20Gb/s
Group2 - Will transmit at 5Gb/s
Group3 - Will transmit at 15Gb/s
Group4 - Will have no BW to use since no BW guarantee was set on it while other groups do have such settings.
```

Privileged VFs

In case a malicious driver is running over one of the VFs, and in case that VF's permissions are not restricted, this may open security holes. However, VFs can be marked as trusted and can thus

receive an exclusive subset of physical function privileges or permissions. For example, in case of allowing all VFs, rather than specific VFs, to enter a promiscuous mode as a privilege, this will enable malicious users to sniff and monitor the entire physical port for incoming traffic, including traffic targeting other VFs, which is considered a severe security hole.

Privileged VFs Configuration

In the configuration example below, the VM is located on VF-0 and has the following MAC address: 11:22:33:44:55:66.
There are two ways to enable or disable trust:

1. Use the standard IP link commands - available from Kernel 4.5 and above.
    a. To enable trust for a specific VF, run:

    ```
    ip link set ens785f1 vf 0 trust on
    ```

    b. To disable trust for a specific VF, run:

    ```
    ip link set ens785f1 vf 0 trust off
    ```

2. Specify echo "ON" or "OFF" to the file located under /sys/class/net/<ETH_IF_NAME> / device/ sriov/<VF index>/trust.
    a. To enable trust for a specific VF, run:

    ```
    echo "ON" > /sys/class/net/ens785f1/device/sriov/0/trust
    ```

    b. To disable trust for a specific VF, run:

    ```
    echo "OFF" > /sys/class/net/ens785f1/device/sriov/0/trust
    ```

Probed VFs

Probing Virtual Functions (VFs) after SR-IOV is enabled might consume the adapter cards' resources. Therefore, it is recommended not to enable probing of VFs when no monitoring of the VM is needed. VF probing can be disabled in two ways, depending on the kernel version installed on your server:

1. If the kernel version installed is v4.12 or above, it is recommended to use the PCI sysfs interface `sriov_drivers_autoprobe`. For more information, see linux-next branch.
2. If the kernel version installed is older than v4.12, it is recommended to use the mlx5_core module parameter probe_vf with driver version 4.1 or above.

Example:

```
echo 0 > /sys/module/mlx5_core/parameters/probe_vf
```

For more information on how to probe VFs, see HowTo Configure and Probe VFs on mlx5 Drivers Community post.

3.3.4.1.5.6 VF Promiscuous Rx Modes

VF Promiscuous Mode

VFs can enter a promiscuous mode that enables receiving the unmatched traffic and all the multicast traffic that reaches the physical port in addition to the traffic originally targeted to the VF. The unmatched traffic is any traffic's DMAC that does not match any of the VFs' or PFs' MAC addresses.
Note: Only privileged/trusted VFs can enter the VF promiscuous mode.

To set the promiscuous mode on for a VF, run:

```
ifconfig eth2 promisc
```

To exit the promiscuous mode, run:

```
ifconfig eth2 -promisc
```

VF All-Multi Mode

VFs can enter an all-multi mode that enables receiving all the multicast traffic sent from/to the other functions on the same physical port in addition to the traffic originally targeted to the VF.
Note: Only privileged/trusted VFs can enter the all-multi RX mode.

To set the all-multi mode on for a VF, run:

```
ifconfig eth2 allmulti
```

To exit the all-multi mode, run:

```
#ifconfig eth2 -allmulti
```

3.3.4.1.6 Uninstalling the SR-IOV Driver

To uninstall SR-IOV driver, perform the following:

1. For Hypervisors, detach all the Virtual Functions (VF) from all the Virtual Machines (VM) or stop the Virtual Machines that use the Virtual Functions.

Please be aware that stopping the driver when there are VMs that use the VFs, will cause machine to hang.

2. Run the script below. Please be aware, uninstalling the driver deletes the entire driver's file, but does not unload the driver.

```
[root@sw1022 ~]# /usr/sbin/ofed_uninstall.sh
This program will uninstall all OFED packages on your machine.
Do you want to continue?[y/N]:y
Running /usr/sbin/vendor_pre_uninstall.sh
Removing OFED Software installations
Running /bin/rpm -e --allmatches kernel-ib kernel-ib-devel libibverbs libibverbs-devel libibverbs-
devel-static libibverbs-utils libmlx4 libmlx4-devel libibcm libibcm-devel libibumad libibumad-devel
libibumad-static libibmad libibmad-devel libibmad-static librdmacm librdmacm-utils librdmacm-devel ibacm
opensm-libs opensm-devel perftest compat-dapl compat-dapl-devel dapl dapl-devel dapl-devel-static dapl-
utils srptools infiniband-diags-guest ofed-scripts opensm-devel
warning: /etc/infiniband/openib.conf saved as /etc/infiniband/openib.conf.rpmsave
Running /tmp/2818-ofed_vendor_post_uninstall.sh
```

3. Restart the server.

3.3.4.1.7 SR-IOV Live Migration

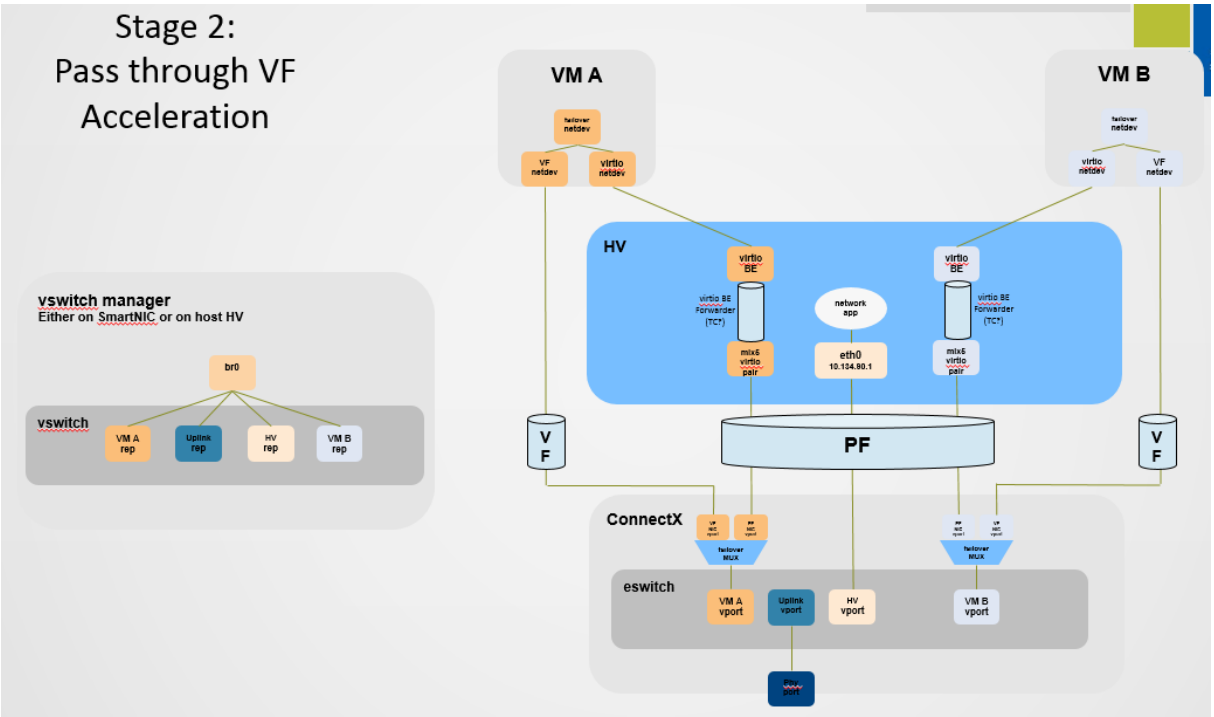
Support for this feature is at beta level.

3.3.4.1.7.1 Overview

This section describes how to set up and perform live migration on VMs with SR-IOV and with actively running traffic.

The below are the requirements for working with SR-IOV Live Migration.

- VM network persistency for applications - VM's applications must survive the migration process
- No internal VM admin configuration
- Support for ASAP^2 solution (kernel OVS hardware offload)
- No use of physical function (PF) network device in Paravirtual (PV) path for failover network traffic
- Use of sub-function (SF) in HyperVisor as failover PV path



3.3.4.1.7.2 Prerequisites

- ConnectX-5 or higher adapter cards
- Hypervisor host with RedHat/CentOS minimal version of 8.0 with MLNX_OFED minimal version of 5.4
- VMs that run on CentOS v8.0 or higher
- Hypervisor host with latest libvirt from hhttps://libvirt.org/news.html#v6-1-0-2020-03-03
- Hypervisor host with latest QEMU from https://www.qemu.org/2021/04/30/qemu-6-0-0/

This section consists of the following steps.

1. Host Servers Configuration.
2. VM OS Installation Using "virt-manager".
3. VFs to VMs Deployment.
4. ASAP^2 with OVS Deployment.
5. Live Migration with Paravirtual Path and Traffic.

3.3.4.1.7.3 Host Servers Configuration

The following steps should be performed in both host servers.

1. Install RedHat/CentOS v8.0 on the host server.
   The CentOS 8.0 ISO image can be downloaded via this link.
2. Connect the host servers to the Ethernet switch.
   Two host servers HV1 (eth0: 10.20.1.118) and HV2 (eth0: 10.20.1.119) connected via an Ethernet switch with switch ports configured/enabled VLAN. For example, vlan=100

3. Install the latest MLNX_OFED version.
   Download and install NVIDIA MLNX_OFED driver for distribution RHEL/CentOS 8.0.

```
# mount -o loop MLNX_OFED_LINIx-5.4-rhel8.0-x86_64.iso /mnt
# cd /mnt
# ./mlnxofedinstall
# reboot
```

4. Configure the host server and NVIDIA NIC with SR-IOV as instructed here.
5. Configure the host server and NVIDIA NIC with sub-function as instructed here (github.com/Mellanox/scalablefunctions/wiki).

```
mlxconfig -d 0000:03:00.0 s PF_BAR2_ENABLE=0 \
      PER_PF_NUM_SF=1 PF_TOTAL_SF=64 PF_SF_BAR_SIZE=8

Set the number of SFs same as that of VFs, as one SF-VF pair is used to attach to one VM.
```

6. Configure storage for VMs' images as shared.
   The default location for VMs' images is /var/lib/libvirt/images, which is a shared location that is set up as an NFS directory in this PoC. For example:

```
# mount <nfs-server>:/opt/nfs/images /var/lib/libvirt/images
```

7. Set up the network bridge "installation" for VMs to enable external communication.
   VMs must be able to download and install extra required packages for external sources.

```
# cd /etc/sysconfig/network-scripts
# vim ifcfg-installation

    DEVICE=installation
    TYPE=Bridge
    BOOTPROTO=dhcp
    ONBOOT=yes
# vim ifcfg-eth0
    BRIDGE=installation
# systemctl network restart
```

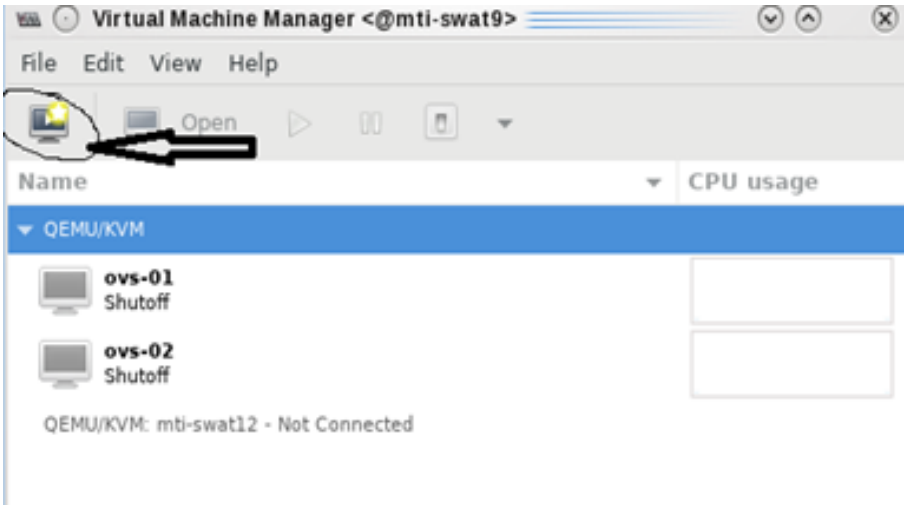
8. Download CentOS v8.0 ISO image for VM installation.
   Download CentOS v8.0 ISO image from one of the mirror sites to the local host.

```
# wget
http://isoredirect.centos.org/centos/8/isos/x86_64/CentOS-8-x86_64-1905-dvd1.iso
```

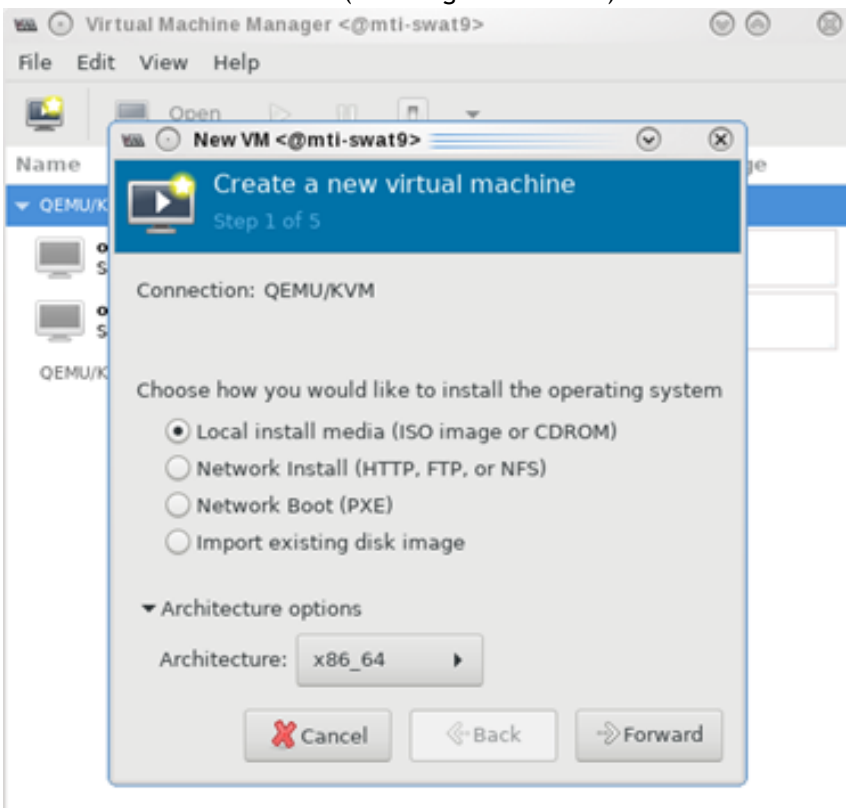
3.3.4.1.7.4 VM OS installation Using "virt-manager"

1. Launch "virt-manager" to create a new VM. Click the icon as shown below.

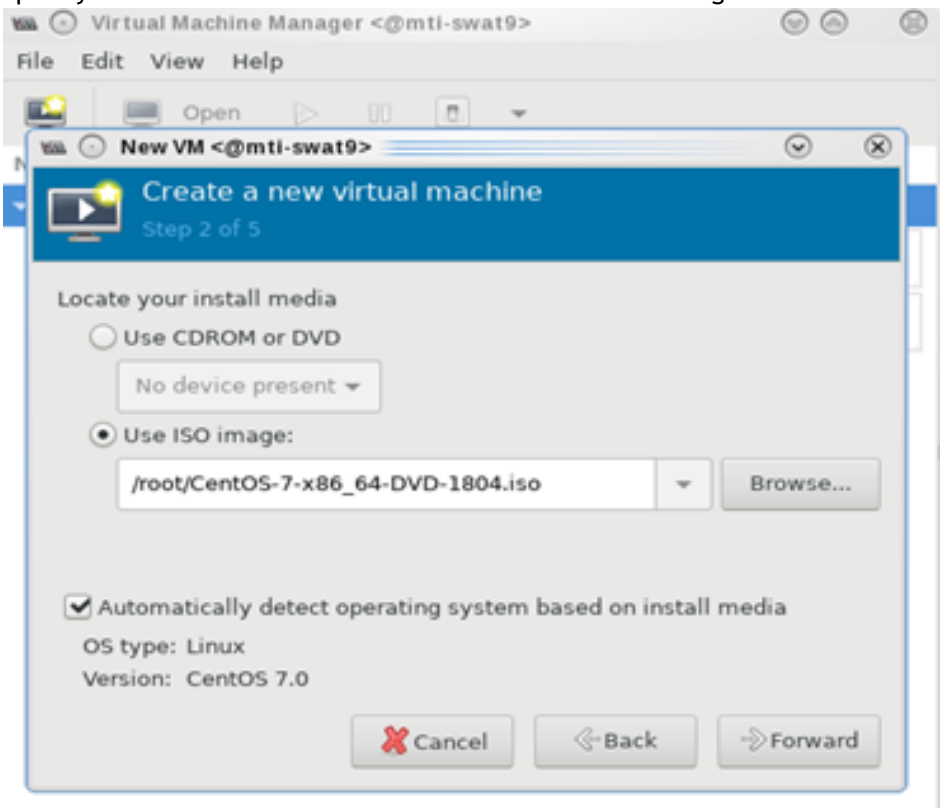
```
# virt-manager
```



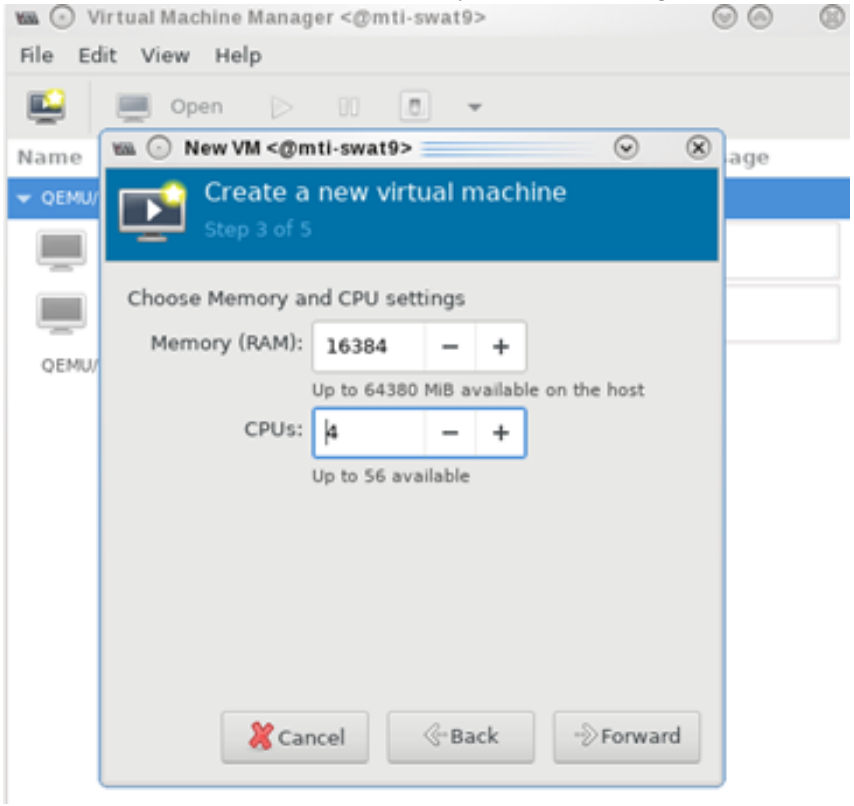
2. Choose “Local install media (ISO images or CDROM)”.



3. Specify the location of the downloaded CentOS 8.0 ISO image.

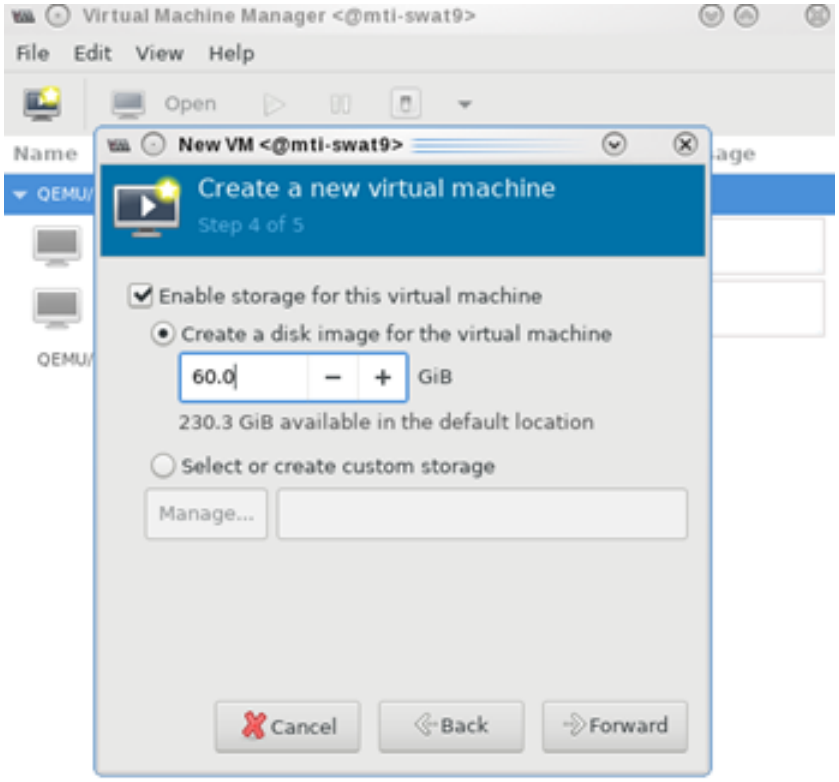


4. Fill in the fields under "Choose Memory and CPU settings" for the VM.



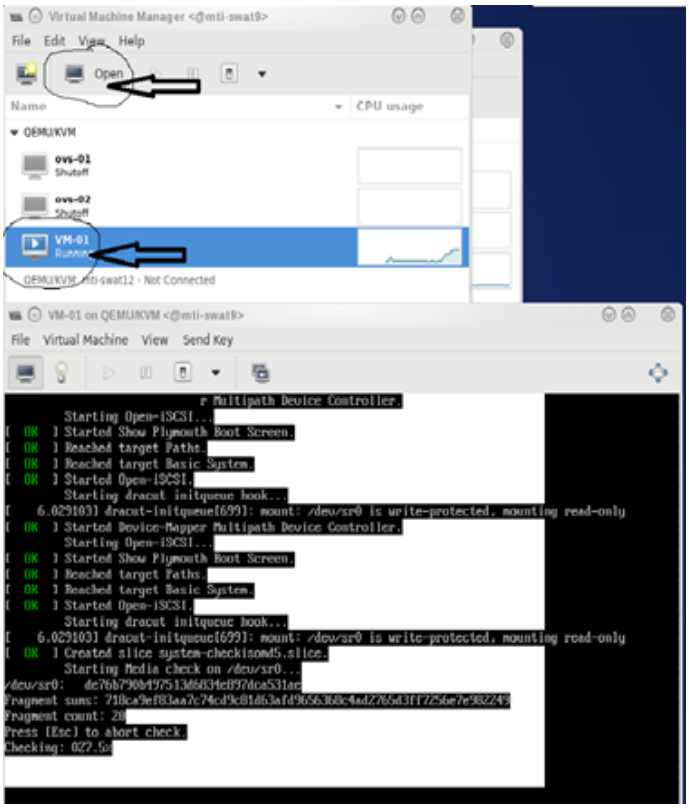
5. Create the disk image at the default root user location, for example: /var/lib/libvirt/images (NFS mount point). Make sure the storage is higher than 120GB and the virtual disk image is

60GB.

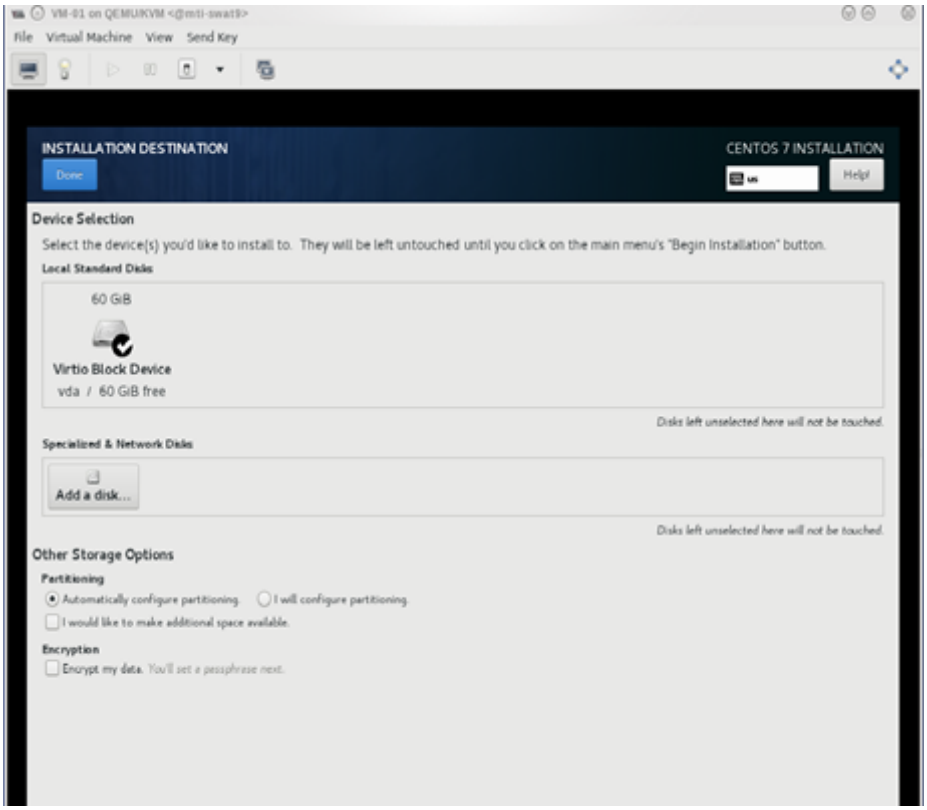


6. In the VM Name field, add "vm-01", and for the network selection, choose "Bridge installation: Host device eth0".

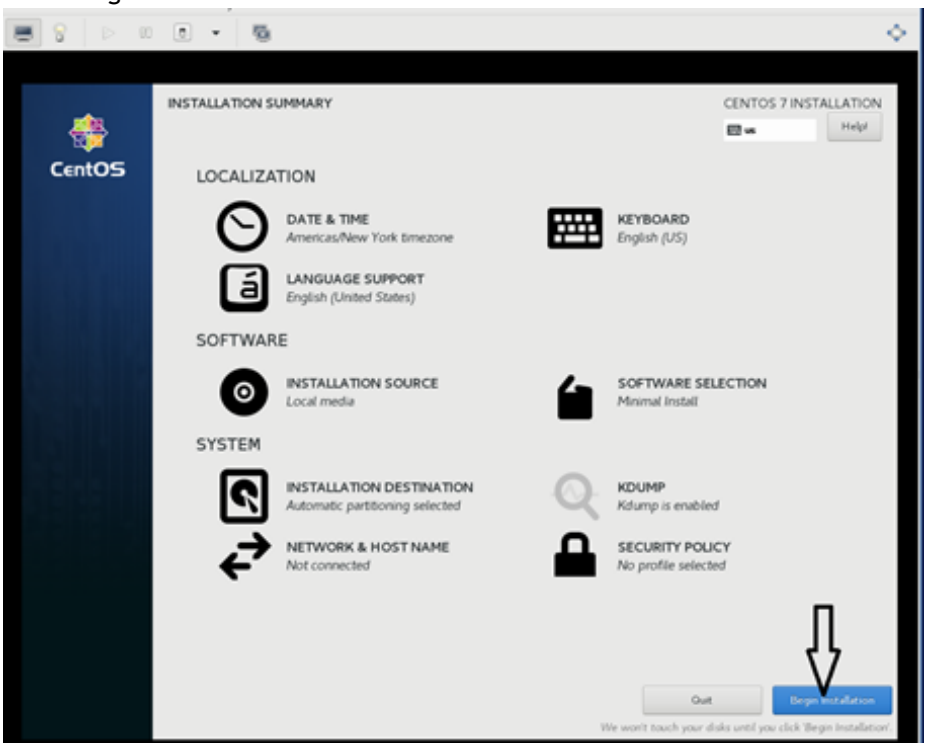
7. Click "vm-01", then "Open".



8. Follow the installation with “Minimal Install” and “Virtual Block Device” selection.



9. Click "Begin Installation".

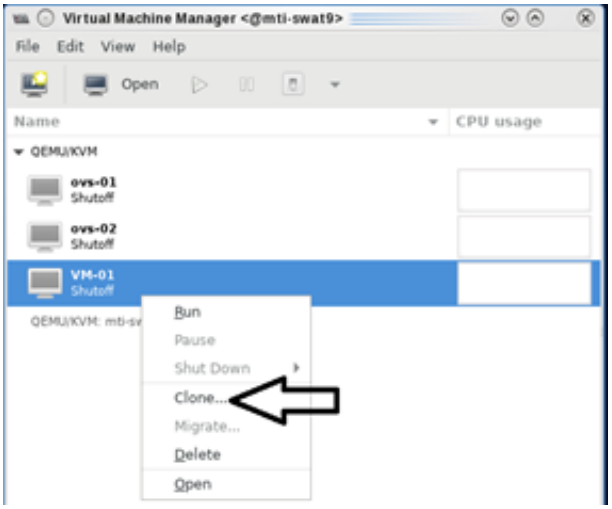


10. Reboot VM "vm-01" after installation is completed.
11. Use the VM's console terminal to enable external communication.

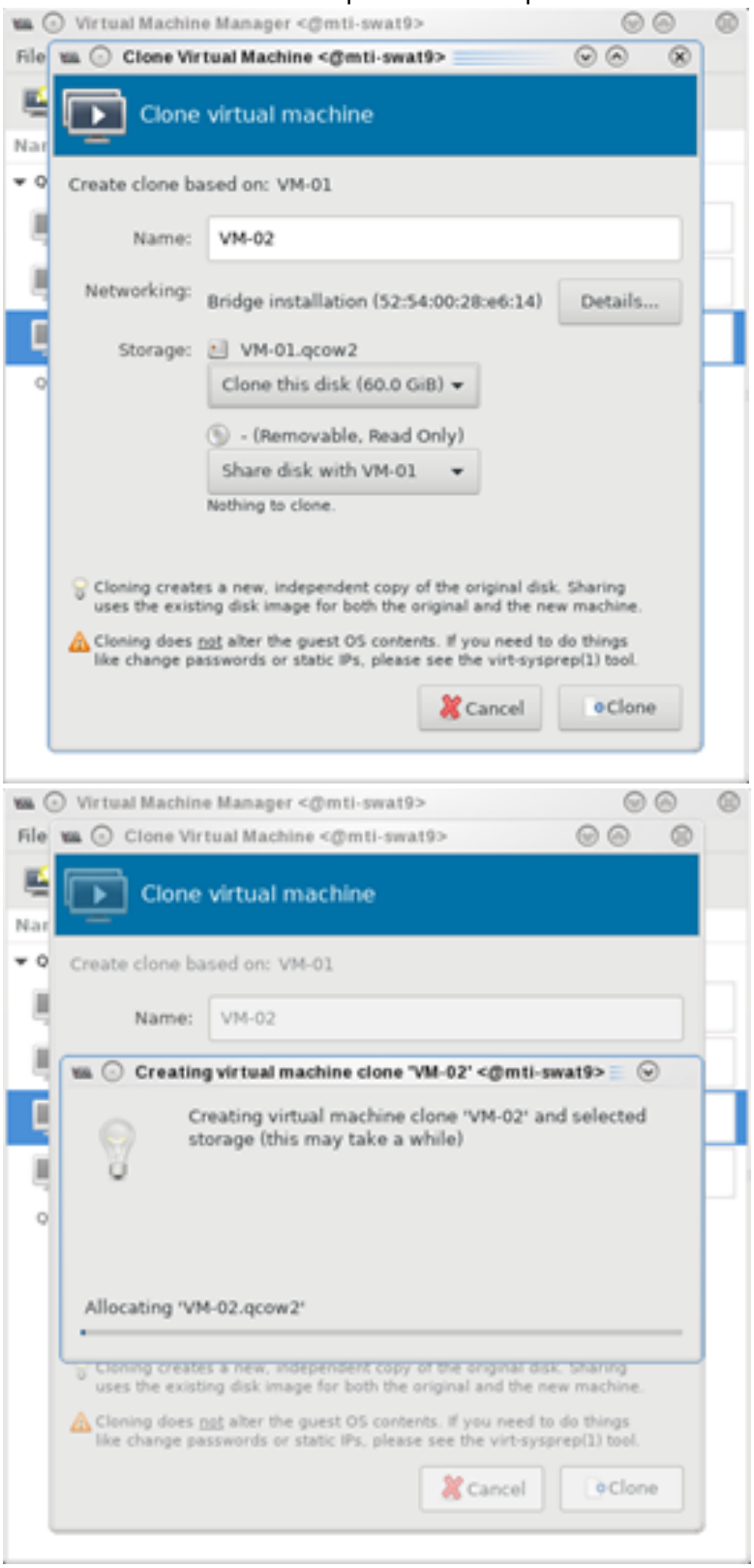
```
# [~]$ vi /etc/sysconfig/network-scripts/ifcfg-eth0
    ONBOOT=yes
    BOOTPROTO=dhcp

# systemctl network restart
```

12. Shut down VM "vm-01" and clone it to VM "vm-02".



13. Clone the virtual disk VM-01.qcow to VM-02.qcow.



14. Test the VM installation and migration without VFs.
    a. Boot both VMs and run ping to each other.

```
[root@vm-01]# ping 10.20.4.8

[root@vm-02]# ping 10.20.4.20
```

b. Perform live migration using the "virsh" command line on HV1 where VM "vm-01" resides.

```
[root@HV1]# virsh list --all
 Id    Name                           State
----------------------------------------------------
 24    VM-01                          running
```

c. Perform live migration.

```
[root@HV1]# virsh migrate --live --unsafe --persistent --verbose VM-01 qemu+ssh://HV2/system
```

d. Verify that vm-01 is migrated and resides at HV2.

```
[root@HV2]# virsh list --all
 Id    Name                           State
----------------------------------------------------
 1     VM-01                          running
 2     VM-02                          running
```

3.3.4.1.7.5 VFs to VMs Deployment

1. Make sure SR-IOV is enabled and VFs are available.

```
[root@HV2]# lspci -D | grep Mellanox
0000:06:00.0 Ethernet controller: Mellanox Technologies MT27800 Family [ConnectX-5]
0000:06:00.1 Ethernet controller: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function]
0000:06:00.2 Ethernet controller: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function]
0000:06:00.3 Ethernet controller: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function]
0000:06:00.4 Ethernet controller: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function]
```

Enable the usage of VF2 "0000:06:00.3" and VF3 "0000:06:00.4" to assign to VM-01 and VM-02 respectively.

2. Attach VF to VM with XML file using "virsh" command line.
   a. Create VM-01-vf.xml and VM-02-vf.xml files to assign VF1 to VM-01 and VF2 to VM-02 as "hostdev" with MAC-address assigned.

```
root@HV1]# cat VM-01-vf.xml
<interface type='hostdev' managed='yes'>
 <mac address='52:54:00:53:53:53'/>
 <source>
 <address type='pci' domain='0x0000' bus='0x06' slot='0x00' function='0x1'/>
 </source>
 <address type='pci' domain='0x0000' bus='0x00' slot='0x0c' function='0x0'/>
</interface>
```

```
[root@HV2]# cat VM-02-vf.xml
<interface type='hostdev' managed='yes'>
 <mac address='52:54:00:54:54:54'/>
 <source>
 <address type='pci' domain='0x0000' bus='0x06' slot='0x00' function='0x2'/>
 </source>
 <address type='pci' domain='0x0000' bus='0x00' slot='0x0c' function='0x0'/>
</interface>
```

   b. Assign the VF to the VM by running the "virsh" command line.
      i. Before attaching the VF, VM-01 and VM-02 should have a single network interface.

```
[root@VM-02]# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT
group default qlen 1000
    link/ether 52:54:00:06:83:8a brd ff:ff:ff:ff:ff:ff
```

ii. On HV1 host, assign VF1 to VM-01.

```
[root@HV1]# virsh attach-device VM-01 VM-01-vf.xml
Device attached successfully
```

iii. On HV2 host, assign VF2 to VM-02.

```
[root@HV2]# virsh attach-device VM-02 VM-02-vf.xml
Device attached successfully
```

iv. After attaching the VFs, VM-01 and VM-02 should have two network interfaces. The second interface "ens12" is the VF with the MAC-address assigned.

```
[root@VM-02]# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT
group default qlen 1000
    link/ether 52:54:00:06:83:8a brd ff:ff:ff:ff:ff:ff
3: ens12: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group
default qlen 1000
    link/ether 52:54:00:54:54:54 brd ff:ff:ff:ff:ff:ff
```

c. Connect to VMs console, configure the IP address of the VF network interface and run traffic.
Install iperf, configure the IP address and run traffic between them on both VMs.

```
[root@VM-01]# yum -y install iperf3
…
Resolving Dependencies
--> Running transaction check
---> Package iperf3.x86_64 0:3.1.7-2.el7 will be installed
…

[root@VM-01]# ip addr add 11.11.11.1 dev ens12


[root@VM-02]# yum -y install iperf3
…
Resolving Dependencies
--> Running transaction check
---> Package iperf3.x86_64 0:3.1.7-2.el7 will be installed
…

[root@VM-02]# ip addr add 11.11.11.2 dev ens12

[root@VM-02]# ping 11.11.11.1
64 bytes from 11.11.11.1: icmp_seq=1 ttl=64 time=0.046 ms
64 bytes from 11.11.11.1: icmp_seq=2 ttl=64 time=0.039 ms

[root@VM-02]# iperf3 -s -f m
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

```
[root@VM-01]# iperf3 -c 11.11.11.2 -P 4 -t 600 -i 1
```

3.3.4.1.7.6 NVIDIA ASAP^2 with OVS Deployment

Perform the NVIDIA ASAP^2 installation and configuration on both HV1 and HV2.

1. Download, build and install OpenvSwitch-2.12.0.

```
root@HV1]# wget https://www.openvswitch.org/releases/openvswitch-2.15.0.tar.gz
[root@HV1]# tar -xzf openvswitch-2.15.0.tar.gz
[root@HV1]# cd openvswitch-2.15.0
[root@HV1]# ./boot.sh
[root@HV1]# ./configure
[root@HV1]# make -j32; make install
[root@HV1]# export PATH=$PATH:/usr/local/share/openvswitch/scripts
```

2. Configure OVS with a single vSwitch "ovs-sriov" with `hw-offload=true` and `tc-policy=verbose`.
    a. Create OVS "ovs-sriov" and set `hw-offload=true` and `tc-policy=verbose`

```
[root@HV1]# export PATH=$PATH:/usr/local/share/openvswitch/scripts
 [root@HV1]# ovs-ctl start

 [root@HV1]# ovs-vsctl add-br ovs-sriov
 [root@HV1]# ovs-vsctl set Open_vSwitch . other_config:hw-offload=true
 [root@HV1]# ovs-vsctl set Open_vSwitch . other_config:tc-policy=verbose
 [root@HV1]# ovs-vsctl set Open_vSwitch . other_config:max-idle=100000
 [root@HV1]# ovs-ctl restart
 [root@HV1]# ovs-vsctl get Open_vSwitch . other_config
{hw-offload="true", tc-policy=verbose}
```

    b. Enable SR-IOV and SWITCHDEV mode by executing "asap_config.sh" script for PF port 1.

```
[root@HV1] cat asap_config.sh
 echo "Configure ASAP & VSWITCH OFFLOAD"
 devlink dev eswitch set pci/0000:${pci}.0 mode switchdev
 ethtool -K $pf hw-tc-offload on
 ip link set dev $pf up

 # Number of Virtual Functions NUM_VFS=2

 # Mellanox NIC ID
 HCA=MT27800

 pci=$(lspci |grep Mellanox|grep $HCA |head -n1|awk '{print $1}'| sed s/\.0\$//g)
pf=$(ls -l /sys/class/net/| grep $pci|awk '{print $9}'| head -n1)
echo "pci=$pci pf=$pf HCA=$HCA"
sh -c "echo $NUM_VFS > /sys/class/net/${pf}/device/sriov_numvfs"

 lspci|grep Mell
 ls -l /sys/class/net/
 ovs-vsctl show
 ovs-dpctl show
```

    c. Create a sub-function on PF port 1 with the script "create_sf.sh".

```
[root@HV1] cat create_sf.sh

 # Number of Sub-Functions
 NUM_SFS=2

 HCA=MT27800

 pci=$(lspci |grep Mellanox|grep "$HCA" |head -n1|awk '{print $1}'| sed s/\.0\$//g)
 DEV=0000:$pci.0

 for ((i=1; i<=$NUM_SFS; i++)); do
 /opt/mellanox/iproute2/sbin/mlxdevm port add pci/0000:${pci}.0 \
        flavour pcisf pfnum 0 sfnum $i
 /opt/mellanox/iproute2/sbin/mlxdevm port function set en6s0f0pf0sf$i state active
done

 [root@HV1] ./create_sf.sh
 [root@HV1] ip link
 …
 47: en6s0f0pf0sf0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group
default qlen 1000
    link/ether 7e:94:e9:79:48:0b brd ff:ff:ff:ff:ff:ff
 48: enp6s0f0s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group
default qlen 1000
    link/ether 36:76:c1:f9:3e:5d brd ff:ff:ff:ff:ff:ff
 49: en6s0f0pf0sf1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group
default qlen 1000
    link/ether 96:03:d2:f1:27:f6 brd ff:ff:ff:ff:ff:ff
 50: enp6s0f0s1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group
default qlen 1000
    link/ether 62:f0:68:a5:79:cb brd ff:ff:ff:ff:ff:ff

NOTES:
======
There are 2 new SF netdevices (enp6s0f0s0, enp6s0f0s1) and their representors netdevices
(en6s0f0pf0sf0, en6s0f0pf0sf1) created
```

    d. Rename the sub-function's netdevices.

```
Rename sub function's netdevices so that both source and destination systems can have same name.

[root@HV1]
$ ip link set eth0 down
$ ip link set eth0 name meth0
$ ip link set meth0 up

$ ip link set eth1 down
$ ip link set eth1 name meth1
$ ip link set meth1 up

$ ip link set eth2 down
$ ip link set eth2 name meth2
$ ip link set meth2 up

$ ip link set eth3 down
$ ip link set eth3 name meth3
$ ip link set meth3 up
```

3.3.4.1.7.7 Live Migration with Paravirtual Path and Traffic

1. Create bonding devices of VF and sub-function (SF) representors.

```
NOTES:
=======
bond0 is bond device of sf1's representor (primary slave) and vf0's representor
bond1 is bond device of sf2's representor (primary slave) and vf1's representor

[root@HV1]# ./bond_setup.sh bond0 en6s0f0pf0sf0 ens2_0
[root@HV1]# ./bond_setup.sh bond1 en6s0f0pf0sf1 ens2_1

[root@HV1]# cat ./bond_setup.sh

BOND=$1

# put here two SF/VF reps for which you want to share the block
SFR1=$2
VFR2=$3

tc qdisc del dev $BOND ingress
tc qdisc del dev $SFR1 ingress
tc qdisc del dev $VFR2 ingress

ip link set dev $SFR1 nomaster
ip link set dev $VFR2 nomaster
ip link del $BOND

ip link add name $BOND type bond

ip link set dev $SFR1 down
ip link set dev $VFR2 down

ip link set dev $BOND type bond mode active-backup
ip link set dev $SFR1 master $BOND
ip link set dev $VFR2 master $BOND

# make SFR1 the primary - this is paravirtual path
echo $SFR1 > /sys/class/net/$BOND/bonding/primary

ip link set dev $SFR1 up
ip link set dev $VFR2 up
ip link set dev $BOND up
```

2. Add Uplink "ens2" and bonding devices to "ovs-sriov" bridge.

```
[root@HV1]# ovs-vsctl add-port ovs-sriov ens2
[root@HV1]# ovs-vsctl add-port ovs-sriov bond0 tag=100
[root@HV1]# ovs-vsctl add-port ovs-sriov bond1 tag=100
```

3. Modify the VM-01's xml file to have the default SF's macvtap-based virtio netdevice.
    a. Edit VM-01 configuration with the same MAC address assigned to VF-1.
    b. Make sure the alias name has the prefix "ua-".

```
Insert below configuration to VM-01.

[root@HV1]# virsh edit VM-01

<interface type='direct'>
   <mac address='52:54:00:53:53:53'/>
   <source dev='enp6s0f0s0' mode='passthrough'/>
   <target dev='macvtap0'/>
   <model type='virtio'/>
   <teaming type='persistent'/>
   <alias name='ua-net0'/>
   <address type='pci' domain='0x0000' bus='0x05' slot='0x00' function='0x0'/>
</interface>
```

c. Edit VM-02 configuration with the same MAC address assigned to VF-2.
d. Make sure the alias name has the prefix "ua-".

```
Insert below configuration to VM-01.

[root@HV1]# virsh edit VM-02

<interface type='direct'>
    <mac address='52:54:00:54:54:54'/>
    <source dev='enp6s0f0s1' mode='passthrough'/>
    <target dev='macvtap1'/>
    <model type='virtio'/>
    <alias name='ua-net1'/>
    <address type='pci' domain='0x0000' bus='0x06' slot='0x00' function='0x0'/>
</interface>
```

4. Restart the VMs and verify that the PV path exists in both VMs and that it is accessible. Each VM should have two extra netdevs, such as: eth0, eth1 where eth0 is master and eth1 is automatically enslaved to eth0.

```
[root@VM-01] ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT qlen 1000
    link/ether 52:54:00:1c:91:0a brd ff:ff:ff:ff:ff:ff
3: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether 52:54:00:53:53:53 brd ff:ff:ff:ff:ff:ff
4: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop master eth0 state DOWN mode DEFAULT qlen 1000
    link/ether 52:54:00:53:53:53 brd ff:ff:ff:ff:ff:ff
```

a. Configure the IP address and run iperf on VMs over SF PF path.

```
[root@VM-01]# ip addr add 11.11.11.1 dev eth0
[root@VM-01]# iperf3 -s -f m
```

```
[root@VM-02]# ip addr add 11.11.11.2 dev eth0
[root@VM-02]# ping 11.11.11.1
[root@VM-02]# iperf3 -c 11.11.11.1 -P 4 -t 600 -i 1
```

b. Modify the XML file of the VFs to link to the persistent device of the SFs.

```
[root@HV1]# cat VM-01-vf.xml
<interface type='hostdev' managed='yes'>
 <mac address='52:54:00:53:53:53'/>
 <source>
 <address type='pci' domain='0x0000' bus='0x06' slot='0x00' function='0x1'/>
 </source>
 <teaming type='transient' persistent='ua-net0'/>
 <address type='pci' domain='0x0000' bus='0x00' slot='0x0c' function='0x0'/>
</interface>
```

```
[root@HV2]# cat VM-02-vf.xml
<interface type='hostdev' managed='yes'>
 <mac address='52:54:00:54:54:54'/>
 <source>
 <address type='pci' domain='0x0000' bus='0x06' slot='0x00' function='0x2'/>
 </source>
 <teaming type='transient' persistent='ua-net1'/>
 <address type='pci' domain='0x0000' bus='0x00' slot='0x0c' function='0x0'/>
</interface>
```

c. Attach VFs to VMs VM-01 and VM-02, and switch to the direct path in HyperVisors HV1 and HV2. I/O traffic should continue after the VFs have been successfully attached to the VMs.

```
[root@HV1] virsh attach-device VM-01 VM-01-vf.xml; echo ens2_0 > /sys/class/net/bond0/bonding/
active_slave
Device attached successfully

[root@HV2] virsh attach-device VM-02 VM-02-vf.xml; echo ens2_1 > /sys/class/net/bond1/bonding/
active_slave
```

```
Device attached successfully
```

Each VM should have one extra netdev from the attached VF that is automatically enslaved to eth0.

```
[root@VM-01] ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT qlen
1000
    link/ether 52:54:00:1c:91:0a brd ff:ff:ff:ff:ff:ff
3: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether 52:54:00:53:53:53 brd ff:ff:ff:ff:ff:ff
4: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop master eth0 state DOWN mode DEFAULT qlen 1000
    link/ether 52:54:00:53:53:53 brd ff:ff:ff:ff:ff:ff
5: ens12: <BROADCAST,MULTICAST> mtu 1500 qdisc noop master eth0 state DOWN mode DEFAULT qlen 1000
    link/ether 52:54:00:53:53:53 brd ff:ff:ff:ff:ff:ff
```

5. Detach VF from VM, switch to SF PV path in HyperVisors HV1 and HV2. I/O traffic should pause 0.5s and then resume.

```
root@HV1] virsh detach-device VM-01 VM-01-vf.xml; echo en6s0f0pf0sf0 > /sys/class/net/bond0/bonding/
active_slave
Device detached successfully
```

6. Perform Live Migration on VM-01. iperf traffic should run as usual.

```
[root@HV1] virsh migrate --live --unsafe --persistent --verbose VM-01 qemu+ssh://HV2/system
```

7. Attach VF to VM again and switch to the direct path in HyperVisor. I/O traffic should run as usual.

```
[root@HV2] virsh attach-device VM-01 VM-01-vf.xml; echo ens2_0 > /sys/class/net/bond0/bonding/active_slave
Device attached successfully
```

3.3.4.2 Enabling Paravirtualization

To enable Paravirtualization:

The example below works on RHEL7.* without a Network Manager.

1. Create a bridge.

```
vim /etc/sysconfig/network-scripts/ifcfg-bridge0
DEVICE=bridge0
TYPE=Bridge
IPADDR=12.195.15.1
NETMASK=255.255.0.0
BOOTPROTO=static
ONBOOT=yes
NM_CONTROLLED=no
DELAY=0
```

2. Change the related interface (in the example below bridge0 is created over eth5).

```
DEVICE=eth5
BOOTPROTO=none
STARTMODE=on
HWADDR=00:02:c9:2e:66:52
TYPE=Ethernet
NM_CONTROLLED=no
ONBOOT=yes
```

```
BRIDGE=bridge0
```

3. Restart the service network.
4. Attach a bridge to VM.

```
ifconfig -a
…
eth6      Link encap:Ethernet  HWaddr 52:54:00:E7:77:99
          inet addr:13.195.15.5  Bcast:13.195.255.255  Mask:255.255.0.0
          inet6 addr: fe80::5054:ff:fee7:7799/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:481 errors:0 dropped:0 overruns:0 frame:0
          TX packets:450 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:22440 (21.9 KiB)  TX bytes:19232 (18.7 KiB)
          Interrupt:10 Base address:0xa000
…
```

3.3.4.3 VXLAN Hardware Stateless Offloads

VXLAN technology provides scalability and security challenges solutions. It requires extension of the traditional stateless offloads to avoid performance drop. ConnectX family cards offer the following stateless offloads for a VXLAN packet, similar to the ones offered to non-encapsulated packets. VXLAN protocol encapsulates its packets using outer UDP header.

Available hardware stateless offloads:

- Checksum generation (Inner IP and Inner TCP/UDP)
- Checksum validation (Inner IP and Inner TCP/UDP)
- TSO support for inner TCP packets
- RSS distribution according to inner packets attributes
- Receive queue selection - inner frames may be steered to specific QPs

3.3.4.3.1 Enabling VXLAN Hardware Stateless Offloads

VXLAN offload is enabled by default for ConnectX-4 family devices running the minimum required firmware version and a kernel version that includes VXLAN support.

To confirm if the current setup supports VXLAN, run:

```
ethtool -k $DEV | grep udp_tnl
```

Example:

```
ethtool -k ens1f0 | grep udp_tnl
tx-udp_tnl-segmentation: on
```

ConnectX-4 family devices support configuring multiple UDP ports for VXLAN offload. Ports can be added to the device by configuring a VXLAN device from the OS command line using the "ip" command.

Note: If you configure multiple UDP ports for offload and exceed the total number of ports supported by hardware, then those additional ports will still function properly, but will not benefit from any of the stateless offloads.

Example:

```
ip link add vxlan0 type vxlan id 10 group 239.0.0.10 ttl 10 dev ens1f0 dstport 4789
ip addr add 192.168.4.7/24 dev vxlan0
ip link set up vxlan0
```

Note: dstport' parameters are not supported in Ubuntu 14.4.

The VXLAN ports can be removed by deleting the VXLAN interfaces.

Example:

```
ip link delete vxlan0
```

3.3.4.3.2 Important Note

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)

3.3.4.4 Q-in-Q Encapsulation per VF in Linux (VST)

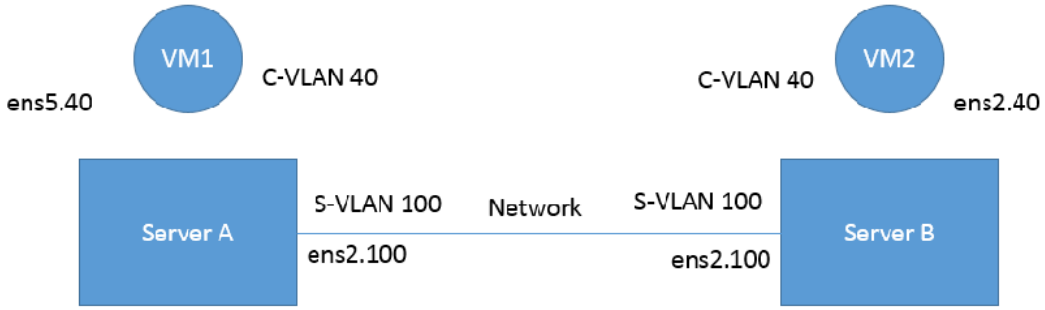
This feature is supported on ConnectX-5 and ConnectX-6 adapter cards only.

ConnectX-4 and ConnectX-4 Lx adapter cards support 802.1Q double-tagging (C-tag stacking on C-tag), refer to "802.1Q Double-Tagging" section.

This section describes the configuration of IEEE 802.1ad QinQ VLAN tag (S-VLAN) to the hypervisor per Virtual Function (VF). The Virtual Machine (VM) attached to the VF (via SR- IOV) can send traffic with or without C-VLAN. Once a VF is configured to VST QinQ encapsulation (VST QinQ), the adapter's hardware will insert S-VLAN to any packet from the VF to the physical port. On the receive side, the adapter hardware will strip the S-VLAN from any packet coming from the wire to that VF.

3.3.4.4.1 Setup

The setup assumes there are two servers equipped with ConnectX-5/ConnectX-6 adapter cards.



3.3.4.4.2 Prerequisites

- Kernel must be of v3.10 or higher, or custom/inbox kernel must support vlan-stag
- Firmware version 16/20.21.0458 or higher must be installed for ConnectX-5/ConnectX-6 HCAs
- The server should be enabled in SR-IOV and the VF should be attached to a VM on the hypervisor.
    - In order to configure SR-IOV in Ethernet mode for ConnectX-5/ConnectX-6 adapter cards, please refer to "Configuring SR-IOV for ConnectX-4/ConnectX-5 (Ethernet)" section. In the following configuration example, the VM is attached to VF0.
- Network Considerations - the network switches may require increasing the MTU (to support 1522 MTU size) on the relevant switch ports.

3.3.4.4.3 Configuring Q-in-Q Encapsulation per Virtual Function for ConnectX-5/ConnectX-6

1. Add the required S-VLAN (QinQ) tag (on the hypervisor) per port per VF. There are two ways to add the S-VLAN:
    a. By using sysfs:

```
echo '100:0:802.1ad' > /sys/class/net/ens1f0/device/sriov/0/vlan
```

    b. By using the ip link command (available only when using the latest Kernel version):

```
ip link set dev ens1f0 vf 0 vlan 100 proto 802.1ad
```

    Check the configuration using the ip link show command:

```
# ip link show ens1f0
 ens1f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT qlen 1000
    link/ether ec:0d:9a:44:37:84 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, vlan 100, vlan protocol 802.1ad, spoof checking off, link-state
auto, trust off
    vf 1 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 2 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 3 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 4 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
```

2. **Optional:** Add S-VLAN priority. Use the qos parameter in the ip link command (or sysfs):

```
ip link set dev ens1f0 vf 0 vlan 100 qos 3 proto 802.1ad
```

Check the configuration using the ip link show command:

```
# ip link show ens1f0
ens1f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT qlen 1000
    link/ether ec:0d:9a:44:37:84 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, vlan 100, qos 3, vlan protocol 802.1ad, spoof checking off, link-state
auto, trust off
    vf 1 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 2 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 3 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 4 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
```

3. Create a VLAN interface on the VM and add an IP address.

```
ip link add link ens5 ens5.40 type vlan protocol 802.1q id 40
ip addr add 42.134.135.7/16 brd 42.134.255.255 dev ens5.40
ip link set dev ens5.40 up
```

4. To verify the setup, run ping between the two VMs and open Wireshark or tcpdump to capture the packet.

3.3.4.5 802.1Q Double-Tagging

This section describes the configuration of 802.1Q double-tagging support to the hypervisor per Virtual Function (VF). The Virtual Machine (VM) attached to the VF (via SR-IOV) can send traffic with or without C-VLAN. Once a VF is configured to VST encapsulation, the adapter's hardware will insert C-VLAN to any packet from the VF to the physical port. On the receive side, the adapter hardware will strip the C-VLAN from any packet coming from the wire to that VF.

3.3.4.5.1 Configuring 802.1Q Double-Tagging per Virtual Function

1. Add the required C-VLAN tag (on the hypervisor) per port per VF. There are two ways to add the C-VLAN:
    a. By using sysfs:

```
echo '100:0:802.1q' > /sys/class/net/ens1f0/device/sriov/0/vlan
```

    b. By using the ip link command (available only when using the latest Kernel version):

```
ip link set dev ens1f0 vf 0 vlan 100
```

Check the configuration using the ip link show command:

```
# ip link show ens1f0
 ens1f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT qlen 1000
    link/ether ec:0d:9a:44:37:84 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, vlan 100, spoof checking off, link-state auto, trust off
    vf 1 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 2 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 3 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 4 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
```

2. Create a VLAN interface on the VM and add an IP address.

```
# ip link add link ens5 ens5.40 type vlan protocol 802.1q id 40
# ip addr add 42.134.135.7/16 brd 42.134.255.255 dev ens5.40
# ip link set dev ens5.40 up
```

3. To verify the setup, run ping between the two VMs and open Wireshark or tcpdump to capture the packet.

3.3.4.6 Scalable Functions

Scalable function is a lightweight function that has a parent PCI function on which it is deployed. Scalable functions are useful for containers where netdevice and RDMA devices of a scalable function can be assigned to a container. This way, the container can get complete offload capabilities of an eswitch, isolation and dedicated accelerated network device. For Step-by-Step Configuration instructions, follow the User Guide here.

3.3.5 Resiliency

The chapter contains the following sections:

- Reset Flow

3.3.5.1 Reset Flow

Reset Flow is activated by default. Once a "fatal device" error is recognized, both the HCA and the software are reset, the ULPs and user application are notified about it, and a recovery process is performed once the event is raised.

Currently, a reset flow can be triggered by a firmware assert with Recover Flow Request (RFR) only. Firmware RFR support should be enabled explicitly using mlxconfig commands.

To query the current value, run:

```
mlxconfig -d /dev/mst/mt4115_pciconf0 query | grep SW_RECOVERY_ON_ERRORS
```

To enable RFR bit support, run:

```
mlxconfig -d /dev/mst/mt4115_pciconf0 set SW_RECOVERY_ON_ERRORS=true
```

3.3.5.1.1 Kernel ULPs

Once a "fatal device" error is recognized, an IB_EVENT_DEVICE_FATAL event is created, ULPs are notified about the incident, and outstanding WQEs are simulated to be returned with "flush in error"

message to enable each ULP to close its resources and not get stuck via calling its "remove_one" callback as part of "Reset Flow".
Once the unload part is terminated, each ULP is called with its "`add_one`" callback, its resources are re-initialized and it is re-activated.

3.3.5.1.2 User Space Applications (IB/RoCE)

Once a "fatal device" error is recognized an IB_EVENT_DEVICE_FATAL event is created, applications are notified about the incident and relevant recovery actions are taken.
Applications that ignore this event enter a zombie state, where each command sent to the kernel is returned with an error, and no completion on outstanding WQEs is expected.
The expected behavior from the applications is to register to receive such events and recover once the above event is raised. Same behavior is expected in case the NIC is unbounded from the PCI and later is rebounded. Applications running over RDMA CM should behave in the same manner once the RDMA_CM_EVENT_DEVICE_REMOVAL event is raised.
The below is an example of using the unbind/bind for NIC defined by "0000:04:00.0"

```
echo 0000:04:00.0 > /sys/bus/pci/drivers/mlx5_core/unbind
echo 0000:04:00.0 > /sys/bus/pci/drivers/mlx5_core/bind
```

3.3.5.1.3 SR-IOV

If the Physical Function recognizes the error, it notifies all the VFs about it by marking their communication channel with that information, consequently, all the VFs and the PF are reset.
If the VF encounters an error, only that VF is reset, whereas the PF and other VFs continue to work unaffected.

3.3.5.1.4 Forcing the VF to Reset

If an outside "reset" is forced by using the PCI sysfs entry for a VF, a reset is executed on that VF once it runs any command over its communication channel.
For example, the below command can be used on a hypervisor to reset a VF defined by 0000:04:00.1:

```
echo 1 >/sys/bus/pci/devices/0000:04:00.1/reset
```

3.3.5.1.5 Extended Error Handling (EEH)

Extended Error Handling (EEH) is a PowerPC mechanism that encapsulates AER, thus exposing AER events to the operating system as EEH events.
The behavior of ULPs and user space applications is identical to the behavior of AER.

3.3.5.1.6 CRDUMP

CRDUMP feature allows for taking an automatic snapshot of the device CR-Space in case the device's FW/HW fails to function properly.

Snapshots Triggers:

The snapshot is triggered after firmware detects a critical issue, requiring a recovery flow.

This snapshot can later be investigated and analyzed to track the root cause of the failure. Currently, only the first snapshot is stored, and is exposed using a temporary virtual file. The virtual file is cleared upon driver reset.
When a critical event is detected, a message indicating CRDUMP collection will be printed to the Linux log. User should then back up the file pointed to in the printed message. The file location format is: /proc/driver/mlx5_core/crdump/<pci address>

Snapshot should be copied by Linux standard tool for future investigation.

3.3.5.1.7 Firmware Tracer

This mechanism allows for the device's FW/HW to log important events into the event tracing system (/sys/kernel/debug/tracing) without requiring any NVIDIA tool.

> ⚠ To be able to use this feature, trace points must be enabled in the kernel.

This feature is enabled by default, and can be controlled using sysfs commands.

➤ *To disable the feature:*

```
echo 0 > /sys/kernel/debug/tracing/events/mlx5/fw_tracer/enable
```

➤ *To enable the feature:*

```
echo 1 > /sys/kernel/debug/tracing/events/mlx5/fw_tracer/enable
```

➤ *To view FW traces using vim text editor:*

```
vim /sys/kernel/debug/tracing/trace
```

3.3.6 Docker Containers

On Linux, Docker uses resource isolation of the Linux kernel, to allow independent "containers" to run within a single Linux kernel instance.
Docker containers are supported on MLNX_OFED using Docker runtime. Virtual RoCE and InfiniBand devices are supported using SR-IOV mode.

Currently, RDMA/RoCE devices are supported in the modes listed in the following table.

Linux Containers Networking Modes

Orchestration and Clustering Tool	Version	Networking Mode	Link Layer	Virtualization Mode
Docker	Docker Engine 17.03 or higher	SR-IOV using sriov-plugin along with docker run wrapper tool	InfiniBand and Ethernet	SR-IOV
Kubernetes	Kubernetes 1.10.3 or higher	SR-IOV using device plugin, and using SR- IOV CNI plugin	InfiniBand and Ethernet	SR-IOV
		VXLAN using IPoIB bridge	InfiniBand	Shared HCA

3.3.6.1 Docker Using SR-IOV

In this mode, Docker engine is used to run containers along with SR-IOV networking plugin. To isolate the virtual devices, docker_rdma_sriov tool should be used. This mode is applicable to both InfiniBand and Ethernet link layers.
To obtain the plugin, visit: hub.docker.com/r/rdma/sriov-plugin
To install the docker_rdma_sriov tool, use the container tools installer available via hub.docker.com/r/rdma/container_tools_installer
For instructions on how to use Docker with SR-IOV, refer to Docker RDMA SRIOV Networking with ConnectX4/ConnectX5/ConnectX6 Community post.

3.3.6.2 Kubernetes Using SR-IOV

In order to use RDMA in Kubernetes environment with SR-IOV networking mode, two main components are required:

1. RDMA device plugin - this plugin allows for exposing RDMA devices in a Pod
2. SR-IOV CNI plugin - this plugin provisions VF net device in a Pod

When used in SR-IOV mode, this plugin enables SR-IOV and performs necessary configuration including setting GUID, MAC, privilege mode, and Trust mode.
The plugin also allocates the VF devices when Pods are scheduled and requested by Kubernetes framework.

3.3.6.3 Kubernetes with Shared HCA

One RDMA device (HCA) can be shared among multiple Pods running in a Kubernetes worker nodes. User defined networks are created using VXLAN or VETH networking devices. RDMA device (HCA) can be shared among multiple Pods running in a Kubernetes worker nodes.

3.3.7 HPC-X

For information on HPC-X®, please refer to HPC-X User Manual at developer.nvidia.com/networking/hpc-x.

3.3.8 Fast Driver Unload

This feature enables optimizing mlx5 driver teardown time in shutdown and kexec flows.

The fast driver unload is disabled by default. To enable it, the `prof_sel` module parameter of mlx5_core module should be set to 3.

3.3.9 OVS Offload Using ASAP² Direct

3.3.9.1 Overview

⚠ Supported on ConnectX-5 and above adapter cards.

Open vSwitch (OVS) allows Virtual Machines (VMs) to communicate with each other and with the outside world. OVS traditionally resides in the hypervisor and switching is based on twelve tuple matching on flows. The OVS software based solution is CPU intensive, affecting system performance and preventing full utilization of the available bandwidth.
NVIDIA Accelerated Switching And Packet Processing (ASAP^2) technology allows OVS offloading by handling OVS data-plane in ConnectX-5 onwards NIC hardware (Embedded Switch or eSwitch) while maintaining OVS control-plane unmodified. As a result, we observe significantly higher OVS performance without the associated CPU load.

As of v5.0, OVS-DPDK became part of MLNX_OFED package. OVS-DPDK supports ASAP^2 just as the OVS-Kernel (Traffic Control (TC) kernel-based solution) does, yet with a different set of features.

The traditional ASAP^2 hardware data plane is built over SR-IOV virtual functions (VFs), so that the VF is passed through directly to the VM, with the NVIDIA driver running within the VM. An alternate approach that is also supported is vDPA (vhost Data Path Acceleration). vDPA allows the connection to the VM to be established using VirtIO, so that the data-plane is built between the SR-IOV VF and the standard VirtIO driver within the VM, while the control-plane is managed on the host by the vDPA application. Two flavors of vDPA are supported, Software vDPA; and Hardware vDPA. Software vDPA management functionality is embedded into OVS-DPDK, while Hardware vDPA uses a standalone application for management, and can be run with both OVS-Kernel and OVS-DPDK. For further information, please see sections VirtIO Acceleration through VF Relay (Software vDPA) and VirtIO Acceleration through Hardware vDPA.

3.3.9.2 Installing OVS-Kernel ASAP² Packages

Install the required packages. For the complete solution, you need to install supporting MLNX_OFED (v4.4 and above), iproute2, and openvswitch packages.

3.3.9.3 Installing OVS-DPDK ASAP² Packages

Run:

```
./mlnxofedinstall --ovs-dpdk -upstream-libs
```

3.3.9.4 Setting Up SR-IOV

Note that this section applies to both OVS-DPDK and OVS-Kernel similarly.

To set up SR-IOV:

1. Choose the desired card.
   The example below shows a dual-ported ConnectX-5 card (device ID 0x1017) and a single SR-IOV VF (Virtual Function, device ID 0x1018).
   In SR-IOV terms, the card itself is referred to as the PF (Physical Function).

```
# lspci -nn | grep Mellanox

0a:00.0 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5] [15b3:1017]
0a:00.1 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5] [15b3:1017]

0a:00.2 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5 Virtual Function]
[15b3:1018]
```

   Enabling SR-IOV and creating VFs is done by the firmware upon admin directive as explained in Step 5 below.

2. Identify the NVIDIA NICs and locate net-devices which are on the NIC PCI BDF.

```
# ls -l /sys/class/net/ | grep 04:00

lrwxrwxrwx 1 root root 0 Mar 27 16:58 enp4s0f0 -> ../../devices/pci0000:00/0000:00:03.0/0000:04:00.0/net/
enp4s0f0
lrwxrwxrwx 1 root root 0 Mar 27 16:58 enp4s0f1 -> ../../devices/pci0000:00/0000:00:03.0/0000:04:00.1/net/
enp4s0f1
lrwxrwxrwx 1 root root 0 Mar 27 16:58 eth0 -> ../../devices/pci0000:00/0000:00:03.0/0000:04:00.2/net/eth0
lrwxrwxrwx 1 root root 0 Mar 27 16:58 eth1 -> ../../devices/pci0000:00/0000:00:03.0/0000:04:00.3/net/eth1
```

   The PF NIC for port #1 is enp4s0f0, and the rest of the commands will be issued on it.

3. Check the firmware version.
   Make sure the firmware versions installed are as state in the Release Notes document.

```
# ethtool -i enp4s0f0 | head -5
driver: mlx5_core
version: 5.0-5
firmware-version: 16.21.0338
expansion-rom-version:
bus-info: 0000:04:00.0
```

4. Make sure SR-IOV is enabled on the system (server, card).
   Make sure SR-IOV is enabled by the server BIOS, and by the firmware with up to N VFs, where N is the number of VFs required for your environment. Refer to "NVIDIA Firmware Tools" below for more details.

```
# cat /sys/class/net/enp4s0f0/device/sriov_totalvfs
4
```

5. Turn ON SR-IOV on the PF device.

```
# echo 2 > /sys/class/net/enp4s0f0/device/sriov_numvfs
```

6. Provision the VF MAC addresses using the IP tool.

```
# ip link set enp4s0f0 vf 0 mac e4:11:22:33:44:50
# ip link set enp4s0f0 vf 1 mac e4:11:22:33:44:51
```

7. Verify the VF MAC addresses were provisioned correctly and SR-IOV was turned ON.

```
# cat /sys/class/net/enp4s0f0/device/sriov_numvfs
2

# ip link show dev enp4s0f0
256: enp4s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master ovs-system state UP mode DEFAULT
group default qlen 1000
    link/ether e4:1d:2d:60:95:a0 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC e4:11:22:33:44:50, spoof checking off, link-state auto
    vf 1 MAC e4:11:22:33:44:51, spoof checking off, link-state auto
```

   In the example above, the maximum number of possible VFs supported by the firmware is 4 and only 2 are enabled.
8. Provision the PCI VF devices to VMs using PCI Pass-Through or any other preferred virt tool of choice, e.g virt-manager.

For further information on SR-IOV, refer to HowTo Configure SR-IOV for ConnectX-4/ConnectX-5/ ConnectX-6 with KVM (Ethernet).

3.3.9.5 OVS Hardware Offloads Configuration

3.3.9.5.1 OVS-Kernel Hardware Offloads

3.3.9.5.1.1 SwitchDev Configuration

1. Unbind the VFs.

```
echo 0000:04:00.2 > /sys/bus/pci/drivers/mlx5_core/unbind
echo 0000:04:00.3 > /sys/bus/pci/drivers/mlx5_core/unbind
```

   ⚠ VMs with attached VFs must be powered off to be able to unbind the VFs.

2. Change the eSwitch mode from Legacy to SwitchDev on the PF device.
   This will also create the VF representor netdevices in the host OS.

```
# devlink dev eswitch set pci/0000:3b:00.0 mode switchdev
```

   ⚠ Before changing the mode, make sure that all VFs are unbound.

   ⚠ To go back to SR-IOV legacy mode, run:
   `# devlink dev eswitch set pci/0000:3b:00.0 mode legacy`
   This will also remove the VF representor netdevices.

   On old OSs or kernels that do not support Devlink, moving to SwitchDev mode can be done using sysfs.

```
# echo switchdev > /sys/class/net/enp4s0f0/compat/devlink/mode
```

3. At this stage, VF representors have been created. To map representor to its VF, make sure to obtain the representor's switchid and portname from:

```
# ip -d link show eth4
41: enp0s8f0_1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
 qlen 1000
    link/ether ba:e6:21:37:bc:d4 brd ff:ff:ff:ff:ff:ff promiscuity 0 addrgenmode eui64 numtxqueues 10
 numrxqueues 10 gso_max_size 65536 gso_max_segs 65535 portname pf0vf1 switchid f4ab580003a1420c
```

   switchid - used to map representor to device, both device PFs have the same switchid.
   portname - used to map representor to PF and VF, value returned is pfXvfY, where X is the PF number and Y is the number of VF.
   On old kernels, switchid and portname can be acquired through sysfs:
4. Bind the VFs.

```
echo 0000:04:00.2 > /sys/bus/pci/drivers/mlx5_core/bind
echo 0000:04:00.3 > /sys/bus/pci/drivers/mlx5_core/bind
```

3.3.9.5.1.2 SwitchDev Performance Tuning

SwitchDev performance can be further improved by tuning it.

Steering Mode

OVS-kernel supports two steering modes for rules insertion into hardware.

1. SMFS – Software Managed Flow Steering (as of MLNX_OFED v5.1, this is the default mode)
   Rules are inserted directly to the hardware by the software (driver). This mode is optimized for rules insertion.
2. DMFS – Device Managed Flow Steering
   Rules insertion is done using firmware commands. This mode is optimized for throughput with a small amount of rules in the system.
   The mode can be controlled via sysfs or devlink API in kernels that support it:

```
Sysfs:
# echo smfs > /sys/class/net/<PF netdev>/compat/devlink/steering_mode

Devlink:
# devlink dev param set pci/0000:00:08.0 name flow_steering_mode value "smfs" cmode runtime

Replace smfs param with dmfs for device managed flow steering
```

Notes:

- The mode should be set prior to moving to SwitchDev, by echoing to the sysfs or invoking the devlink command.
- Only when moving to SwitchDev will the driver use the mode set by the previous step.
- Mode cannot be changed after moving to SwitchDev.
- The steering mode is applicable for SwitchDev mode only, meaning it does not affect legacy SR-IOV or other configurations.

mlx5 debugfs was extended to support presenting Software Steering resources: dr_domain including it's tables, matchers and rules.
The interface is read-only.

While dump is being created, new steering rules cannot be inserted/deleted.
The steering information is dumped in the CSV form with the following format:
`<object_type>,<object_ID>, <object_info>,...,<object_info>`

This data can be read at the following path: `/sys/kernel/debug/mlx5/<BDF>/steering/fdb/<domain_handle>`

Example:

```
# cat /sys/kernel/debug/mlx5/0000:82:00.0/steering/fdb/dmn_000018644
3100,0x55caa4621c50,0xee802,4,65533
3101,0x55caa4621c50,0xe0100008
```

You can then use the steering dump parser to make the output more human readable.
The parser can be found in the following public GitHub repository: https://github.com/Mellanox/mlx_steering_dump

vPort Match Mode

OVS-kernel support two modes that define how the rules on match on vport.

1. Metadata – rules match on metadata instead of vport number (default mode).
   This mode is needed in order to support SR-IOV Live migration and Dual port RoCE features.
   Matching on Metadata can have a performance impact.
2. Legacy – rules match on vport number.
   In this mode, performance can be higher in comparison to Metadata. It can still be used only if none of the above features (SR-IOV Live migration and Dual port RoCE) is enabled/used.
   The mode can be controlled via sysfs:

   ```
   Set Legacy:
   # echo legacy > /sys/class/net/<PF netdev>/compat/devlink/vport_match_mode

   Set metadata:
   Devlink:
   # echo metadata > /sys/class/net/<PF netdev>/compat/devlink/vport_match_mode
   ```

   Note: This mode should be set prior to moving to SwitchDev, by echoing to the sysfs.

Flow Table Large Group Number

Offloaded flows, including Connection Tracking, are added to Virtual Switch Forwarding Data Base (FDB) flow tables. FDB tables have a set of flow groups, where each flow group saves the same traffic pattern flows. E.g, for connection tracking offloaded flow, TCP and UDP are different traffic patterns which will end up in two different flow groups.

A flow group has a limited size to save flow entries. As default, the driver has 15 big FDB flow groups. Each of these big flow groups can save 4M / (15 + 1) = 256k different 5-tuple flow entries at most. For scenarios with more than 15 traffic patterns, the driver provides a module parameter (num_of_groups) to allow customization and performance tuning.

The mode can be controlled via module param or devlink API for kernels that support it:

```
Module param:
# echo <num_of_groups> > /sys/module/mlx5_core/parameters/num_of_groups

Devlink:
# devlink dev param set pci/0000:82:00.0 name fdb_large_groups \
      cmode driverinit value 20
```

Notes:

- In MLNX_OFED v5.1, the default value was changed from 4 to 15.
- The change takes effect immediately if there is no flow inside the FDB table (no traffic running and all offloaded flows are aged out). And it can be dynamically changed without reloading the driver.
  If there are still offloaded flows residual when changing this parameter, it will only take effect after all flows have aged out.

3.3.9.5.1.3 Open vSwitch Configuration

Open vSwitch configuration is a simple OVS bridge configuration with SwitchDev.

1. Run the openvswitch service.

   ```
   # systemctl start openvswitch
   ```

2. Create an OVS bridge (here it's named ovs-sriov).

   ```
   # ovs-vsctl add-br ovs-sriov
   ```

3. Enable hardware offload (disabled by default).

   ```
   # ovs-vsctl set Open_vSwitch . other_config:hw-offload=true
   ```

4. Restart the openvswitch service. This step is required for HW offload changes to take effect.

   ```
   # systemctl restart openvswitch
   ```

   HW offload policy can also be changed by setting the tc-policy using one on the following values:

   * none - adds a TC rule to both the software and the hardware (default)

   * skip_sw - adds a TC rule only to the hardware

   * skip_hw - adds a TC rule only to the software

   The above change is used for debug purposes.

5. Add the PF and the VF representor netdevices as OVS ports.

   ```
   # ovs-vsctl add-port ovs-sriov enp4s0f0
   # ovs-vsctl add-port ovs-sriov enp4s0f0_0
   # ovs-vsctl add-port ovs-sriov enp4s0f0_1
   ```

   Make sure to bring up the PF and representor netdevices.

```
# ip link set dev enp4s0f0 up
# ip link set dev enp4s0f0_0 up
# ip link set dev enp4s0f0_1 up
```

The PF represents the uplink (wire).

```
# ovs-dpctl show
system@ovs-system:
        lookups: hit:0 missed:192 lost:1
        flows: 2
        masks: hit:384 total:2 hit/pkt:2.00
        port 0: ovs-system (internal)
        port 1: ovs-sriov (internal)
        port 2: enp4s0f0
        port 3: enp4s0f0_0
        port 4: enp4s0f0_1
```

6. Run traffic from the VFs and observe the rules added to the OVS data-path.

```
# ovs-dpctl dump-flows

recirc_id(0),in_port(3),eth(src=e4:11:22:33:44:50,dst=e4:1d:2d:a5:f3:9d),
eth_type(0x0800),ipv4(frag=no), packets:33, bytes:3234, used:1.196s, actions:2

recirc_id(0),in_port(2),eth(src=e4:1d:2d:a5:f3:9d,dst=e4:11:22:33:44:50),
eth_type(0x0800),ipv4(frag=no), packets:34, bytes:3332, used:1.196s, actions:3
```

In the example above, the ping was initiated from VF0 (OVS port 3) to the outer node (OVS port 2), where the VF MAC is e4:11:22:33:44:50 and the outer node MAC is e4:1d:2d:a5:f3:9d As shown above, two OVS rules were added, one in each direction.
Note that you can also verify offloaded packets by adding type=offloaded to the command. For example:

```
# ovs-appctl dpctl/dump-flows type=offloaded
```

3.3.9.5.1.4 Open vSwitch Performance Tuning

Flow Aging

The aging timeout of OVS is given in ms and can be controlled using the following command.

```
# ovs-vsctl set Open_vSwitch . other_config:max-idle=30000
```

TC Policy

Specifies the policy used with HW offloading.

- none - adds a TC rule to both the software and the hardware (default)
- skip_sw - adds a TC rule only to the hardware
- skip_hw - adds a TC rule only to the software

Example:

```
# ovs-vsctl set Open_vSwitch . other_config:tc-policy=skip_sw
```

Note: TC policy should only be used for debugging purposes.

Max-Revalidator

Specifies the maximum time (in ms) that revalidator threads will wait for kernel statistics before executing flow revalidation.

```
# ovs-vsctl set Open_vSwitch . other_config:max-revalidator=10000
```

n-handler-threads

Specifies the number of threads for software datapaths to use for handling new flows.
The default value is the number of online CPU cores minus the number of revalidators.

```
# ovs-vsctl set Open_vSwitch . other_config:n-handler-threads=4
```

n-revalidator-threads

Specifies the number of threads for software datapaths to use for revalidating flows in the datapath.

```
# ovs-vsctl set Open_vSwitch . other_config:n-revalidator-threads=4
```

vlan-limit

Limits the number of VLAN headers that can be matched to the specified number.

```
# ovs-vsctl set Open_vSwitch . other_config:vlan-limit=2
```

3.3.9.5.1.5 Basic TC Rules Configuration

Offloading rules can also be added directly, and not only through OVS, using the tc utility.

To create an offloading rule using TC:

1. Create an ingress qdisc (queueing discipline) for each interface that you wish to add rules into.

   ```
   # tc qdisc add dev enp4s0f0 ingress
   # tc qdisc add dev enp4s0f0_0 ingress
   # tc qdisc add dev enp4s0f0_1 ingress
   ```

2. Add TC rules using flower classifier in the following format.

   ```
   # tc filter add dev NETDEVICE ingress protocol PROTOCOL prio PRIORITY \
   [chain CHAIN] flower [ MATCH_LIST ] [ action ACTION_SPEC ]
   ```

   Note: List of supported matches (specifications) and actions can be found in Classification Fields (Matches) section.

3. Dump the existing tc rules using flower classifier in the following format.

   ```
   # tc [ -s ] filter show dev NETDEVICE ingress
   ```

3.3.9.5.1.6 SR-IOV VF LAG

SR-IOV VF LAG allows the NIC's physical functions (PFs) to get the rules that the OVS will try to offload to the bond net-device, and to offload them to the hardware e-switch. Bond modes supported are:

- Active-Backup
- XOR
- LACP

SR-IOV VF LAG enables complete offload of the LAG functionality to the hardware. The bonding creates a single bonded PF port. Packets from up-link can arrive from any of the physical ports, and will be forwarded to the bond device.

When hardware offload is used, packets from both ports can be forwarded to any of the VFs. Traffic from the VF can be forwarded to both ports according to the bonding state. Meaning, when in active-backup mode, only one PF is up, and traffic from any VF will go through this PF. When in XOR or LACP mode, if both PFs are up, traffic from any VF will split between these two PFs.

SR-IOV VF LAG Configuration on ASAP^2

To enable SR-IOV VF LAG, both physical functions of the NIC should first be configured to SR-IOV SwitchDev mode, and only afterwards bond the up-link representors.

The example below shows the creation of bond interface on two PFs:

1. Load bonding device and enslave the up-link representor (currently PF) net-device devices.

```
modprobe bonding mode=802.3ad
Ifup bond0 (make sure ifcfg file is present with desired bond configuration)
ip link set enp4s0f0 master bond0
ip link set enp4s0f1 master bond0
```

2. Add the VF representor net-devices as OVS ports. If tunneling is not used, add the bond device as well.

```
ovs-vsctl add-port ovs-sriov bond0
ovs-vsctl add-port ovs-sriov enp4s0f0_0
ovs-vsctl add-port ovs-sriov enp4s0f1_0
```

3. Make sure to bring up the PF and the representor netdevices.

```
ip link set dev bond0 up
ip link set dev enp4s0f0_0 up
ip link set dev enp4s0f1_0 up
```

Once SR-IOV VF LAG is configured, all VFs of the two PFs will become part of the bond, and will behave as described above.

Limitations

- In VF LAG mode, outgoing traffic in load balanced mode is according to the origin ring, thus, half of the rings will be coupled with port 1 and half with port 2. All the traffic on the same ring will be sent from the same port.

- VF LAG configuration is not supported when the NUM_OF_VFS configured in mlxconfig is higher than 64.

Using TC with VF LAG

Both rules can be added using either of the following.

1. Shared block (supported from kernel 4.16 and RHEL/CentOS 7.7 and above).

```
# tc qdisc add dev bond0 ingress_block 22 ingress
# tc qdisc add dev ens4p0 ingress_block 22 ingress
# tc qdisc add dev ens4p1 ingress_block 22 ingress
```

   a. Add drop rule.

```
# tc filter add block 22 protocol arp parent ffff: prio 3 \
    flower \
        dst_mac e4:11:22:11:4a:51 \
        action drop
```

   b. Add redirect rule from bond to representor.

```
# tc filter add block 22 protocol arp parent ffff: prio 3 \
        flower \
        dst_mac e4:11:22:11:4a:50 \
        action mirred egress redirect dev ens4f0_0
```

   c. Add redirect rule from representor to bond.

```
# tc filter add dev ens4f0_0 protocol arp parent ffff: prio 3 \
    flower \
        dst_mac ec:0d:9a:8a:28:42 \
        action mirred egress redirect dev bond0
```

2. Without shared block (supported from kernel 4.15 and below).

   a. Add redirect rule from bond to representor.

```
# tc filter add dev bond0 protocol arp parent ffff: prio 1 \
        flower \
        dst_mac e4:11:22:11:4a:50 \
        action mirred egress redirect dev ens4f0_0
```

   b. Add redirect rule from representor to bond.

```
# tc filter add dev ens4f0_0 protocol arp parent ffff: prio 3 \
        flower \
        dst_mac ec:0d:9a:8a:28:42 \
        action mirred egress redirect dev bond0
```

3.3.9.5.1.7 Classification Fields (Matches)

OVS-Kernel supports multiple classification fields which packets can fully or partially match.

Ethernet Layer 2

- Destination MAC
- Source MAC
- Ethertype

Supported on all kernels.

In OVS dump flows:

```
skb_priority(0/0),skb_mark(0/0),in_port(eth6),eth(src=00:02:10:40:10:0d
,dst=68:54:ed:00:af:de),eth_type(0x8100), packets:1981, bytes:206024, used:0.440s, dp:tc, actions:eth7
```

Using TC rules:

```
tc filter add dev $rep parent ffff: protocol arp pref 1 \
flower \
dst_mac e4:1d:2d:5d:25:35 \
src_mac e4:1d:2d:5d:25:34 \
action mirred egress redirect dev $NIC
```

IPv4/IPv6

- Source address
- Destination address
- Protocol
    - TCP/UDP/ICMP/ICMPv6
- TOS
- TTL (HLIMIT)

Supported on all kernels.

In OVS dump flows:

```
Ipv4:
ipv4(src=0.0.0.0/0.0.0.0,dst=0.0.0.0/0.0.0.0,proto=17,tos=0/0,ttl=0/0,frag=no)
Ipv6:
ipv6(src=::/::,dst=1:1:1::3:1040:1008,label=0/0,proto=58,tclass=0/0x3,hlimit=64),
```

Using TC rules:

```
IPv4:
tc filter add dev $rep parent ffff: protocol ip pref 1 \
flower \
dst_ip 1.1.1.1 \
src_ip 1.1.1.2 \
ip_proto TCP \
ip_tos 0x3 \
ip_ttl 63 \
action mirred egress redirect dev $NIC


IPv6:
tc filter add dev $rep parent ffff: protocol ipv6 pref 1 \
flower \
dst_ip 1:1:1::3:1040:1009 \
src_ip 1:1:1::3:1040:1008 \
ip_proto TCP \
ip_tos 0x3 \
ip_ttl 63\
action mirred egress redirect dev $NIC
```

TCP/UDP Source and Destination ports & TCP Flags

- TCP/UDP source and destinations ports
- TCP flags

Supported kernels are kernel > 4.13 and RHEL > 7.5

In OVS dump flows:

```
TCP: tcp(src=0/0,dst=32768/0x8000),
UDP: udp(src=0/0,dst=32768/0x8000),
```

```
TCP flags: tcp_flags(0/0)
```

Using TC rules:

```
tc filter add dev $rep parent ffff: protocol ip pref 1 \
flower \
ip_proto TCP \
dst_port 100 \
src_port 500 \
tcp_flags 0x4/0x7 \
action mirred egress redirect dev $NIC
```

VLAN

- ID
- Priority
- Inner vlan ID and Priority

Supported kernels: All (QinQ: kernel 4.19 and higher, and RHEL 7.7 and higher)

In OVS dump flows:

```
eth_type(0x8100),vlan(vid=2347,pcp=0),
```

Using TC rules:

```
tc filter add dev $rep parent ffff: protocol 802.1Q pref 1 \
                    flower \
                    vlan_ethtype 0x800 \
                    vlan_id 100 \
                    vlan_prio 0 \
            action mirred egress redirect dev $NIC
QinQ:
tc filter add dev $rep parent ffff: protocol 802.1Q pref 1 \
                    flower \
                    vlan_ethtype 0x8100  \
                    vlan_id 100 \
                    vlan_prio 0 \
                    cvlan_id 20 \
                    cvlan_prio 0 \
                    cvlan_ethtype 0x800 \
            action mirred egress redirect dev $NIC
```

Tunnel

- ID (Key)
- Source IP address
- Destination IP address
- Destination port
- TOS (supported from kernel 4.19 and above & RHEL 7.7 and above)
- TTL (support from kernel 4.19 and above & RHEL 7.7 and above)
- Tunnel options (Geneve)

Supported kernels:

- VXLAN: All
- GRE: Kernel > 5.0, RHEL 7.7 and above
- Geneve: Kernel > 5.0, RHEL 7.7 and above

In OVS dump flows:

```
tunnel(tun_id=0x5,src=121.9.1.1,dst=131.10.1.1,ttl=0/0,tp_dst=4789,flags(+key))
```

Using TC rules:

```
# tc filter add dev $rep protocol 802.1Q parent ffff: pref 1
flower \
vlan_ethtype 0x800 \
vlan_id 100 \
vlan_prio 0 \
action mirred egress redirect dev $NIC
QinQ:
# tc filter add dev vxlan100 protocol ip parent ffff: \
                flower \
                        skip_sw \
                        dst_mac e4:11:22:11:4a:51 \
                        src_mac e4+:11:22:11:4a:50 \
                        enc_src_ip 20.1.11.1 \
                        enc_dst_ip 20.1.12.1 \
                        enc_key_id 100 \
                        enc_dst_port 4789 \
                action tunnel_key unset \
                action mirred egress redirect dev ens4f0_0
```

3.3.9.5.1.8 Supported Actions

Forward

Forward action allows for packet redirection:

- From VF to wire
- Wire to VF
- VF to VF

Supported on all kernels.

In OVS dump flows:

```
skb_priority(0/0),skb_mark(0/0),in_port(eth6),eth(src=00:02:10:40:10:0d
,dst=68:54:ed:00:af:de),eth_type(0x8100), packets:1981, bytes:206024, used:0.440s, dp:tc, actions:eth7
```

Using TC rules:

```
tc filter add dev $rep parent ffff: protocol arp pref 1 \
                        flower \
                        dst_mac e4:1d:2d:5d:25:35 \
                        src_mac e4:1d:2d:5d:25:34 \
                    action mirred egress redirect dev $NIC
```

Drop

Drop action allows to drop incoming packets.

Supported on all kernels.

In OVS dump flows:

```
skb_priority(0/0),skb_mark(0/0),in_port(eth6),eth(src=00:02:10:40:10:0d
,dst=68:54:ed:00:af:de),eth_type(0x8100), packets:1981, bytes:206024, used:0.440s, dp:tc, actions:drop
```

Using TC rules:

```
tc filter add dev $rep parent ffff: protocol arp pref 1 \
                            flower \
                            dst_mac e4:1d:2d:5d:25:35 \
                            src_mac e4:1d:2d:5d:25:34 \
                    action drop
```

Statistics

By default, each flow collects the following statistics:

- Packets – number of packets which hit the flow
- Bytes – total number of bytes which hit the flow
- Last used – the amount of time passed since last packet hit the flow

Supported on all kernels.

In OVS dump flows:

```
skb_priority(0/0),skb_mark(0/0),in_port(eth6),eth(src=00:02:10:40:10:0d
,dst=68:54:ed:00:af:de),eth_type(0x8100), packets:1981, bytes:206024, used:0.440s, dp:tc, actions:drop
```

Using TC rules:

```
#tc -s filter show dev $rep ingress

filter protocol ip pref 2 flower chain 0
filter protocol ip pref 2 flower chain 0 handle 0x2
eth_type ipv4
ip_proto tcp
src_ip 192.168.140.100
src_port 80
skip_sw
in_hw
    action order 1: mirred (Egress Redirect to device p0v11_r) stolen
    index 34 ref 1 bind 1 installed 144 sec used 0 sec
    Action statistics:
    Sent 388344 bytes 2942 pkt (dropped 0, overlimits 0 requeues 0)
    backlog 0b 0p requeues 0
```

Tunnels (Encapsulation/Decapsulation)

OVS-kernel supports offload of tunnels using encapsulation and decapsulation actions.

- Encapsulation – pushing of tunnel header is supported on Tx
- Decapsulation – popping of tunnel header is supported on Rx

Supported Tunnels:

- VXLAN (IPv4/IPv6) – supported on all Kernels
- GRE (IPv4/IPv6) – supported on kernel 5.0 and above & RHEL 7.6 and above
- Geneve (IPv4/IPv6) - supported on kernel 5.0 and above & RHEL 7.6 and above

OVS configuration:
In case of offloading tunnel, the PF/bond should not be added as a port in the OVS datapath. It should rather be assigned with the IP address to be used for encapsulation.
The example below shows two hosts (PFs) with IPs 1.1.1.177 and 1.1.1.75, where the PF device on both hosts is enp4s0f0, and the VXLAN tunnel is set with VNID 98:

- On the first host:

```
# ip addr add 1.1.1.177/24 dev enp4s0f1

# ovs-vsctl add-port ovs-sriov vxlan0 -- set interface vxlan0 type=vxlan
options:local_ip=1.1.1.177 options:remote_ip=1.1.1.75 options:key=98
```

- On the second host:

```
# ip addr add 1.1.1.75/24 dev enp4s0f1
```

```
# ovs-vsctl add-port ovs-sriov vxlan0 -- set interface vxlan0 type=vxlan
options:local_ip=1.1.1.75 options:remote_ip=1.1.1.177 options:key=98
```

- for GRE IPv4 tunnel need use type=gre
- for GRE IPv6 tunnel need use type=ip6gre
- for GENEVE tunnel need use type=geneve

When encapsulating guest traffic, the VF's device MTU must be reduced to allow the host/HW to add the encap headers without fragmenting the resulted packet. As such, the VF's MTU must be lowered by 50 bytes from the uplink MTU for IPv4 and 70 bytes for IPv6.

Tunnel offload using TC rules:

```
Encapsulation:
# tc filter add dev ens4f0_0 protocol 0x806 parent ffff: \
                flower \
                       skip_sw \
                       dst_mac e4:11:22:11:4a:51 \
                       src_mac e4:11:22:11:4a:50 \
                action tunnel_key set \
                src_ip 20.1.12.1 \
                dst_ip 20.1.11.1 \
                id 100 \
                action mirred egress redirect dev vxlan100

Decapsulation:
# tc filter add dev vxlan100 protocol 0x806 parent ffff: \
                flower \
                         skip_sw \
                         dst_mac e4:11:22:11:4a:51 \
                         src_mac e4:11:22:11:4a:50 \
                         enc_src_ip 20.1.11.1 \
                         enc_dst_ip 20.1.12.1 \
                         enc_key_id 100 \
                         enc_dst_port 4789 \
                action tunnel_key unset \
                  action mirred egress redirect dev ens4f0_0
```

VLAN Push/Pop

OVS-kernel supports offload of vlan header push/pop actions.

- Push—pushing of VLAN header is supported on Tx
- Pop—popping of tunnel header is supported on Rx

Starting with ConnectX-6 Dx hardware models and above, pushing of VLAN header is also supported on Rx, and popping of VLAN header is also supported on Tx.

OVS Configuration

Add a tag=$TAG section for the OVS command line that adds the representor ports. For example, VLAN ID 52 is being used here.

```
# ovs-vsctl add-port ovs-sriov enp4s0f0
# ovs-vsctl add-port ovs-sriov enp4s0f0_0 tag=52
# ovs-vsctl add-port ovs-sriov enp4s0f0_1 tag=52
```

The PF port should not have a VLAN attached. This will cause OVS to add VLAN push/pop actions when managing traffic for these VFs.

Dump Flow Example

```
recirc_id(0),in_port(3),eth(src=e4:11:22:33:44:50,dst=00:02:c9:e9:bb:b2),eth_type(0x0800),ipv4(frag=no), \
```

```
packets:0, bytes:0, used:never, actions:push_vlan(vid=52,pcp=0),2

recirc_id(0),in_port(2),eth(src=00:02:c9:e9:bb:b2,dst=e4:11:22:33:44:50),eth_type(0x8100), \
vlan(vid=52,pcp=0),encap(eth_type(0x0800),ipv4(frag=no)), packets:0, bytes:0, used:never, actions:pop_vlan,3
```

VLAN Offload using TC Rules Example

```
# tc filter add dev ens4f0_0 protocol ip parent ffff: \
                flower \
                        skip_sw \
                        dst_mac e4:11:22:11:4a:51 \
                        src_mac e4:11:22:11:4a:50 \
                action vlan push id 100 \
                action mirred egress redirect dev ens4f0

# tc filter add dev ens4f0 protocol 802.1Q parent ffff: \
                flower \
                        skip_sw \
                        dst_mac e4:11:22:11:4a:51 \
                        src_mac e4:11:22:11:4a:50 \
                        vlan_ethtype 0x800 \
                        vlan_id 100 \
                        vlan_prio 0 \
                action vlan pop \
                action mirred egress redirect dev ens4f0_0
```

TC Configuration for ConnectX-6 Dx and Above

Example of VLAN Offloading with popping header on Tx and pushing on Rx using TC Rules:

```
# tc filter add dev ens4f0_0 ingress protocol 802.1Q parent ffff: \
        flower \
                vlan_id 100 \
        action vlan pop \
        action tunnel_key set \
                src_ip 4.4.4.1 \
                dst_ip 4.4.4.2 \
                dst_port 4789 \
                id 42 \
        action mirred egress redirect dev vxlan0

# tc filter add dev vxlan0 ingress protocol all parent ffff: \
        flower \
                enc_dst_ip 4.4.4.1 \
                enc_src_ip 4.4.4.2 \
                enc_dst_port 4789 \
                enc_key_id 42 \
        action tunnel_key unset \
        action vlan push id 100 \
        action mirred egress redirect dev ens4f0_0
```

Header Rewrite

This action allows for modifying packet fields.

Ethernet Layer 2

- Destination MAC
- Source MAC

Supported kernels: Kernel 4.14 and above & RHEL 7.5 and above

In OVS dump flows:

```
skb_priority(0/0),skb_mark(0/0),in_port(eth6),eth(src=00:02:10:40:10:0d
,dst=68:54:ed:00:af:de),eth_type(0x8100), packets:1981, bytes:206024, used:0.440s, dp:tc, actions:
set(eth(src=68:54:ed:00:f4:ab,dst=fa:16:3e:dd:69:c4)),eth7
```

Using TC rules:

```
tc filter add dev $rep parent ffff: protocol arp pref 1 \
                            flower \
                            dst_mac e4:1d:2d:5d:25:35 \
                            src_mac e4:1d:2d:5d:25:34 \
                    action pedit ex \
                    munge eth dst set 20:22:33:44:55:66 \
```

```
                munge eth src set aa:ba:cc:dd:ee:fe \
                action mirred egress redirect dev $NIC
```

IPv4/IPv6

- Source address
- Destination address
- Protocol
- TOS
- TTL (HLIMIT)

Supported kernels: Kernel 4.14 and above & RHEL 7.5 and above

In OVS dump flows:

```
Ipv4:
    set(eth(src=de:e8:ef:27:5e:45,dst=00:00:01:01:01:01)),
    set(ipv4(src=10.10.0.111,dst=10.20.0.122,ttl=63))
Ipv6:
    set(ipv6(dst=2001:1:6::92eb:fcbe:f1c8,hlimit=63)),
```

Using TC rules:

```
IPv4:
tc filter add dev $rep parent ffff: protocol ip pref 1 \
                           flower \
                           dst_ip 1.1.1.1 \
                           src_ip 1.1.1.2 \
                           ip_proto TCP \
                           ip_tos 0x3 \
                           ip_ttl 63 \
                  pedit ex \
                  munge ip src set 2.2.2.1 \
                  munge ip dst set 2.2.2.2 \
                  munge ip tos set 0 \
                  munge ip ttl dec \
                  action mirred egress redirect dev $NIC


IPv6:
tc filter add dev $rep parent ffff: protocol ipv6 pref 1 \
                           flower \
                           dst_ip 1:1:1::3:1040:1009 \
                           src_ip 1:1:1::3:1040:1008 \
                           ip_proto tcp \
                           ip_tos 0x3 \
                           ip_ttl 63\
                  pedit ex \
                  munge ipv6 src set 2:2:2::3:1040:1009 \
                  munge ipv6 dst set 2:2:2::3:1040:1008 \
                  munge ipv6 hlimit dec \
                  action mirred egress redirect dev $NIC
```

⚠ IPv4 and IPv6 header rewrite is only supported with match on UDP/TCP/ICMP protocols.

TCP/UDP Source and Destination Ports

- TCP/UDP source and destinations ports

Supported kernels: kernel > 4.16 & RHEL > 7.6

In OVS dump flows:

```
TCP:

        set(tcp(src= 32768/0xffff,dst=32768/0xffff)),
UDP:
```

```
        set(udp(src= 32768/0xffff,dst=32768/0xffff)),
```

Using TC rules:

```
TCP:

        tc filter add dev $rep parent ffff: protocol ip pref 1 \
                            flower \
                            dst_ip 1.1.1.1 \
                            src_ip 1.1.1.2 \
                            ip_proto tcp \
                            ip_tos 0x3 \
                            ip_ttl 63 \
                    pedit ex \
                    pedit ex munge ip tcp sport set 200
                    pedit ex munge ip tcp dport set 200
                    action mirred egress redirect dev $NIC

UDP:
        tc filter add dev $rep parent ffff: protocol ip pref 1 \
                            flower \
                            dst_ip 1.1.1.1 \
                            src_ip 1.1.1.2 \
                            ip_proto udp \
                            ip_tos 0x3 \
                            ip_ttl 63 \
                    pedit ex \
                    pedit ex munge ip udp sport set 200
                    pedit ex munge ip udp dport set 200
                    action mirred egress redirect dev $NIC
```

VLAN

- ID

Supported on all kernels.

In OVS dump flows:

```
Set(vlan(vid=2347,pcp=0/0)),
```

Using TC rules:

```
tc filter add dev $rep parent ffff: protocol 802.1Q pref 1 \
                    flower \
                    vlan_ethtype 0x800 \
                    vlan_id 100 \
                    vlan_prio 0 \
            action vlan modify id 11 pipe
            action mirred egress redirect dev $NIC
```

Connection Tracking

The TC connection tracking action performs connection tracking lookup by sending the packet to netfilter conntrack module. Newly added connections may be associated, via the ct commit action, with a 32 bit mark, 128 bit label and source/destination NAT values.

The following example allows ingress tcp traffic from the uplink representor to vf1_rep, while assuring that egress traffic from vf1_rep is only allowed on established connections. In addition, mark and source IP NAT is applied.

In OVS dump flows:

```
ct(zone=2,nat)
ct_state(+est+trk)
actions:ct(commit,zone=2,mark=0x4/0xffffffff,nat(src=5.5.5.5))
```

Using TC rules:

```
# tc filter add dev $uplink_rep ingress chain 0 prio 1 proto ip \
                    flower \
                    ip_proto tcp   \
                    ct_state -trk \
                action ct zone 2 nat pipe
                action goto chain 2
  # tc filter add dev $uplink_rep ingress chain 2 prio 1 proto ip \
                     flower \
                     ct_state +trk+new \
                action ct zone 2 commit mark 0xbb nat src addr 5.5.5.7 pipe \
                action mirred egress redirect dev $vf1_rep
 # tc filter add dev $uplink_rep ingress chain 2 prio 1 proto ip \
                    flower \
                    ct_zone 2 \
                    ct_mark 0xbb \
                    ct_state +trk+est \
                 action mirred egress redirect dev $vf1_rep

  #Setup filters on $vf1_rep, allowing only established connections of zone 2   through, and reverse nat (dst nat
in this case)
 # tc filter add dev $vf1_rep ingress chain 0 prio 1 proto ip \
                    flower \
                    ip_proto tcp \
                    ct_state -trk \
                action ct zone 2 nat pipe \
                action goto chain 1
  # tc filter add dev $vf1_rep ingress chain 1 prio 1 proto ip \
                    flower \
                    ct_zone 2 \
                    ct_mark 0xbb \
                    ct_state +trk+est \
                 action mirred egress redirect dev eth0
```

Connection Tracking Performance Tuning

- Max offloaded connections—specifies the limit on the number of offloaded connections. Example:

```
# devlink dev param set pci/${pci_dev} name ct_max_offloaded_conns value $max cmode runtime
```

- Allow mixed NAT/non-NAT CT—allows offloading of the following scenario:

```
•   cookie=0x0, duration=21.843s, table=0, n_packets=4838718, n_bytes=241958846, ct_state=-
trk,ip,in_port=enp8s0f0 actions=ct(table=1,zone=2)
•   cookie=0x0, duration=21.823s, table=1, n_packets=15363, n_bytes=773526,
ct_state=+new+trk,ip,in_port=enp8s0f0 actions=ct(commit,zone=2,nat(dst=11.11.11.11)),output:"enp8s0f0_1"
•   cookie=0x0, duration=21.806s, table=1, n_packets=4767594, n_bytes=238401190,
ct_state=+est+trk,ip,in_port=enp8s0f0 actions=ct(zone=2,nat),output:"enp8s0f0_1"
```

  Example:

```
# echo enable > /sys/class/net/<device>/compat/devlink/ct_action_on_nat_conns
```

Forward to Chain (TC Only)

TC interface supports adding flows on different chains. Only chain 0 is accessed by default. Access to the other chains requires usafe of the goto action.

In this example, a flow is created on chain 1 without any match and redirect to wire.

The second flow is created on chain 0 and match on source MAC and action goto chain 1.

This example simulates simple MAC spoofing.

```
#tc filter add dev $rep parent ffff: protocol all chain 1 pref 1 \
                    flower \
                action mirred egress redirect dev $NIC

#tc filter add dev $rep parent ffff: protocol all chain 1 pref 1 \
                    flower \
                    src_mac aa:bb:cc:aa:bb:cc
                action goto chain 1
```

3.3.9.5.1.9 Port Mirroring (Flow Based VF Traffic Mirroring for ASAP²)

Unlike para-virtual configurations, when the VM traffic is offloaded to the hardware via SR-IOV VF, the host side Admin cannot snoop the traffic (e.g. for monitoring).

ASAP² uses the existing mirroring support in OVS and TC along with the enhancement to the offloading logic in the driver to allow mirroring the VF traffic to another VF.

The mirrored VF can be used to run traffic analyzer (tcpdump, wireshark, etc) and observe the traffic of the VF being mirrored.

The example below shows the creation of port mirror on the following configuration:

```
# ovs-vsctl show
  09d8a574-9c39-465c-9f16-47d81c12f88a
      Bridge br-vxlan
                  Port "enp4s0f0_1"
                     Interface "enp4s0f0_1"
                  Port "vxlan0"
                      Interface "vxlan0"
                                type: vxlan
                                options: {key="100", remote_ip="192.168.1.14"}
                  Port "enp4s0f0_0"
                      Interface "enp4s0f0_0"
                  Port "enp4s0f0_2"
                      Interface "enp4s0f0_2"
                  Port br-vxlan
                      Interface br-vxlan
                                type: internal
      ovs_version: "2.14.1"
```

- To set enp4s0f0_0 as the mirror port, and mirror all of the traffic:

```
# ovs-vsctl -- --id=@p get port enp4s0f0_0 \
            -- --id=@m create mirror name=m0 select-all=true output-port=@p \
            -- set bridge br-vxlan mirrors=@m
```

- To set enp4s0f0_0 as the mirror port, and only mirror the traffic, the destination is enp4s0f0_1:

```
# ovs-vsctl -- --id=@p1 get port enp4s0f0_0 \
            -- --id=@p2 get port enp4s0f0_1 \
            -- --id=@m create mirror name=m0 select-dst-port=@p2 output-port=@p1 \
            -- set bridge br-vxlan mirrors=@m
```

- To set enp4s0f0_0 as the mirror port, and only mirror the traffic the source is enp4s0f0_1:

```
# ovs-vsctl -- --id=@p1 get port enp4s0f0_0 \
            -- --id=@p2 get port enp4s0f0_1 \
            -- --id=@m create mirror name=m0 select-src-port=@p2 output-port=@p1 \
            -- set bridge br-vxlan mirrors=@m
```

- To set enp4s0f0_0 as the mirror port and mirror, all the traffic on enp4s0f0_1:

```
# ovs-vsctl -- --id=@p1 get port enp4s0f0_0 \
      -- --id=@p2 get port enp4s0f0_1 \
      -- --id=@m create mirror name=m0 select-dst-port=@p2 select-src-port=@p2 output-port=@p1 \
      -- set bridge br-vxlan mirrors=@m
```

To clear the mirror port:

```
 # ovs-vsctl clear bridge br-vxlan mirrors
```

Mirroring using TC:

```
Mirror to VF
```

```
tc filter add dev $rep parent ffff: protocol arp pref 1 \
                        flower \
                        dst_mac e4:1d:2d:5d:25:35 \
                        src_mac e4:1d:2d:5d:25:34 \
                        action mirred egress mirror dev $mirror_rep pipe \
                        action mirred egress redirect dev $NIC

Mirror to tunnel:
tc filter add dev $rep parent ffff: protocol arp pref 1 \
                        flower \
                        dst_mac e4:1d:2d:5d:25:35 \
                        src_mac e4:1d:2d:5d:25:34 \
                action tunnel_key set \
                src_ip 1.1.1.1 \
                dst_ip 1.1.1.2 \
                dst_port 4789 \
                id 768 \
                pipe \
                action mirred egress mirror dev vxlan100 pipe \
                action mirred egress redirect dev $NIC
```

3.3.9.5.1.10 Forward to Multiple Destinations

Forward to up 32 destinations (representors and tunnels) is supported using TC.

Example 1: forward to 32 VFs.

```
tc filter add dev $NIC parent ffff: protocol arp pref 1 \
                        flower \
                        dst_mac e4:1d:2d:5d:25:35 \
                        src_mac e4:1d:2d:5d:25:34 \
                        action mirred egress mirror dev $rep0 pipe \
                        action mirred egress mirror dev $rep1 pipe \
...
                        action mirred egress mirror dev $rep30 pipe \
                        action mirred egress redirect dev $rep31
```

Example 2: forward to 16 tunnels.

```
tc filter add dev $rep parent ffff: protocol arp pref 1 \
                        flower \
                        dst_mac e4:1d:2d:5d:25:35 \
                        src_mac e4:1d:2d:5d:25:34 \
                        action tunnel_key set src_ip $ip_src dst_ip $ip_dst \
dst_port 4789 id 0 nocsum \
pipe action mirred egress mirror dev vxlan0 pipe \
                        action tunnel_key set src_ip $ip_src dst_ip $ip_dst \
dst_port 4789 id 1 nocsum \
pipe action mirred egress mirror dev vxlan0 pipe \
...
                        action tunnel_key set src_ip $ip_src dst_ip $ip_dst \
dst_port 4789 id 15 nocsum \
pipe action mirred egress redirect dev vxlan0
```

- TC supports up to 32 actions
- If header rewrite is used, then all destinations should have the same header rewrite
- If VLAN push/pop is used, then all destinations should have the same VLAN ID and actions

3.3.9.5.1.11 sFLOW

This feature allows for monitoring traffic sent between two VMs on the same host using an sFlow collector.

The example below assumes the environment is configured as described below.

```
# ovs-vsctl show
  09d8a574-9c39-465c-9f16-47d81c12f88a
      Bridge br-vxlan
                  Port "enp4s0f0_1"
                    Interface "enp4s0f0_1"
```

```
                Port "vxlan0"
                  Interface "vxlan0"
                            type: vxlan
                            options: {key="100", remote_ip="192.168.1.14"}
                Port "enp4s0f0_0"
                  Interface "enp4s0f0_0"
                Port "enp4s0f0_2"
                  Interface "enp4s0f0_2"
                Port br-vxlan
                  Interface br-vxlan
                            type: internal
    ovs_version: "2.14.1"
```

To sample all traffic over the OVS bridge:

```
# ovs-vsctl -- --id=@sflow create sflow agent=\"$SFLOW_AGENT\" \
             target=\"$SFLOW_TARGET:$SFLOW_PORT\" header=$SFLOW_HEADER \
             sampling=$SFLOW_SAMPLING polling=10 \
             -- set bridge br-vxlan sflow=@sflow
```

Parameter	Description
SFLOW_AGENT	Indicates that the sFlow agent should send traffic from SFLOW_AGENT's IP address
SFLOW_TARGET	Remote IP address of the sFLOW collector
SFLOW_HEADER	Size of packet header to sample (in bytes)
SFLOW_SAMPLING	Sample rate

To clear the sFLOW configuration:

```
# ovs-vsctl clear bridge br-vxlan sflow
```

To list the sFLOW configuration:

```
# ovs-vsctl list sflow
```

sFLOW using TC:

```
Sample to VF
tc filter add dev $rep parent ffff: protocol arp pref 1 \
                        flower \
                        dst_mac e4:1d:2d:5d:25:35 \
                        src_mac e4:1d:2d:5d:25:34 \
                        action sample rate 10 group 5 trunc 96 \
                        action mirred egress redirect dev $NIC
```

Userspace application is needed in order to process to sampled packet from the kernel.
Example: https://github.com/Mellanox/libpsample

3.3.9.5.1.12 Rate Limit

OVS-kernel supports offload of VF rate limit using OVS configuration and TC.

The example below sets a rate limit to the VF related to representor eth0 to 10Mbps.

```
OVS:
```

```
 # ovs-vsctl set interface eth0 ingress_policing_rate=10000

tc:
 # tc_filter add dev eth0 root prio 1 protocol ip matchall skip_sw action police rate 10mbit burst 20k
```

3.3.9.5.1.13 Kernel Requirements

This kernel config should be enabled in order to support switchdev offload.

- CONFIG_NET_ACT_CSUM - needed for action csum
- CONFIG_NET_ACT_PEDIT - needed for header rewrite
- CONFIG_NET_ACT_MIRRED - needed for basic forward
- CONFIG_NET_ACT_CT - needed for connection tracking (supported from kernel 5.6)
- CONFIG_NET_ACT_VLAN - needed for action vlan push/pop
- CONFIG_NET_ACT_GACT
- CONFIG_NET_CLS_FLOWER
- CONFIG_NET_CLS_ACT
- CONFIG_NET_SWITCHDEV
- CONFIG_NET_TC_SKB_EXT - needed for connection tracking (supported from kernel 5.6)
- CONFIG_NET_ACT_CT - needed for connection tracking (supported from kernel 5.6)
- CONFIG_NFT_FLOW_OFFLOAD
- CONFIG_NET_ACT_TUNNEL_KEY
- CONFIG_NF_FLOW_TABLE - needed for connection tracking (supported from kernel 5.6)
- CONFIG_SKB_EXTENSIONS - needed for connection tracking (supported from kernel 5.6)
- CONFIG_NET_CLS_MATCHALL
- CONFIG_NET_ACT_POLICE
- CONFIG_MLX5_ESWITCH

3.3.9.5.1.14 VF Metering

OVS-kernel supports offloading of VF metering (TX and RX) using sysfs. Metering of number of packets per second (PPS) and bytes per second (BPS) is supported.

The example bellow sets Rx meter on VF 0 with value 10Mbps BPS.

```
echo 10000000 > /sys/class/net/enp4s0f0/device/sriov/0/meters/rx/bps/rate
echo 65536 > /sys/class/net/enp4s0f0/device/sriov/0/meters/rx/bps/burst
```

The example bellow sets Tx meter on VF 0 with value 1000 PPS.

```
echo 1000 > /sys/class/net/enp4s0f0/device/sriov/0/meters/tx/pps/rate
echo 100 > /sys/class/net/enp4s0f0/device/sriov/0/meters/tx/pps/burst
```

Both rate and burst must not be zero and burst may need to be adjusted according to the requirements.

The following counters can be used to query the number dropped packet/bytes:

```
#cat /sys/class/net/enp8s0f0/device/sriov/0/meters/rx/pps/packets_dropped
#cat /sys/class/net/enp8s0f0/device/sriov/0/meters/rx/pps/bytes_dropped
#cat /sys/class/net/enp8s0f0/device/sriov/0/meters/rx/bps/packets_dropped
#cat /sys/class/net/enp8s0f0/device/sriov/0/meters/rx/bps/bytes_dropped
```

```
#cat /sys/class/net/enp8s0f0/device/sriov/0/meters/tx/pps/packets_dropped
#cat /sys/class/net/enp8s0f0/device/sriov/0/meters/tx/pps/bytes_dropped
#cat /sys/class/net/enp8s0f0/device/sriov/0/meters/tx/bps/packets_dropped
#cat /sys/class/net/enp8s0f0/device/sriov/0/meters/tx/bps/bytes_dropped
```

3.3.9.5.1.15 Representor Metering

Metering for uplink and VF representors traffic support has been added.

Traffic going to a representor device can be a result of a miss in the embedded switch (eSwitch) FDB tables. This means that a packet which arrived from that representor into the eSwitch was not matched against the existing rules in the hardware FDB tables and needs to be forwarded to software to be handled there and is, therefore, forwarded to the originating representor device driver.

The meter allows to configure the max rate [packets/sec] and max burst [packets] for traffic going to the representor driver. Any traffic exceeding values provided by the user will be dropped in hardware. There are statistics that show number of dropped packets.

The configuration of a representors metering is done via a new sysfs called `miss_rl_cfg`.

- Full path of the miss_rl_cfg parameter: / `sys/class/net//rep_config/miss_rl_cfg`
- Usage: `echo "<rate> <burst>" > /sys/class/net//rep_config/miss_rl_cfg`. Rate is the max rate of packets allowed for this representor (in packets/sec units) and burst is the max burst size allowed for this representor (in packets units). Both values must be specified. The default is 0 for both, meaning unlimited rate and burst.

To view the amount of packets and bytes that were dropped due to traffic exceeding the user-provided rate and burst, two read-only sysfs for statistics are exposed.

- `/sys/class/net//rep_config/miss_rl_dropped_bytes` counts how many FDB-miss bytes were dropped due to reaching the miss limits
- `/sys/class/net//rep_config/miss_rl_dropped_packets` counts how many FDB-miss packets were dropped due to reaching the miss limits

3.3.9.5.1.16 Open vSwitch Metering

There are two types of meters, kpps (kilobits per second) and pktps(packets per second), which are described in Meter Syntax of OpenFlow 1.3+ Switch Meter Table Commands. OVS-Kernel supports offloading them both.

The example below is to offload a kpps meter. Please follow the steps after doing basic configurations as described in section 5.1.3.

1. Create OVS meter with a target rate.

   ```
   ovs-ofctl -O OpenFlow13 add-meter ovs-sriov meter=1,kbps,band=type=drop,rate=204800
   ```

2. Delete the default rule.

   ```
   ovs-ofctl del-flows ovs-sriov
   ```

3. Configure OpenFlow rules. Here VF bandwidth on the receiving side will be limited by the rate configured in step 1.

```
  ovs-ofctl -O OpenFlow13 add-flow ovs-sriov 'ip,dl_dst=e4:11:22:33:44:50,actions=
meter:1,output:enp4s0f0_0'
  ovs-ofctl -O OpenFlow13 add-flow ovs-sriov 'ip,dl_src=e4:11:22:33:44:50,actions= output:enp4s0f0'
  ovs-ofctl -O OpenFlow13 add-flow ovs-sriov 'arp,actions=normal'
```

4. Run iperf server and be ready to receive UDP traffic. On the outer node, run iperf client to send UDP traffic to this VF. After traffic starts, check the offloaded meter rule.

```
 ovs-appctl dpctl/dump-flows --names type=offloaded

recirc_id(0),in_port(enp4s0f0),eth(dst=e4:11:22:33:44:50),eth_type(0x0800),ipv4(frag=no), packets:11626587,
bytes:17625889188, used:0.470s, actions:meter(0),enp4s0f0_0
```

In order to verify metering, iperf client should set the target bandwidth with a number which is larger than the meter rate configured. Then it will be visible that packets are received with the limited rate on the server side and the extra packets are dropped by hardware.

3.3.9.5.1.17 Multiport eSwitch Mode

The multiport eSwitch mode allows to add rules on a VF representor with an action forwarding the packet to the physical port of the physical function. This can be used to implement failover or forward packets based on external information such the cost of the route.

1. To configure this more, the nvconig parameter LAG_RESOURCE_ALLOCATION must be set.
2. After the driver loads, configure multiport eSwitch for each PF where enp8s0f0 and enp8s0f1 represent the netdevices for the PFs.

```
echo multiport_esw > /sys/class/net/enp8s0f0/compat/devlink/lag_port_select_mode
echo multiport_esw > /sys/class/net/enp8s0f1/compat/devlink/lag_port_select_mode
```

   The mode becomes operational after entering switchdev mode on both PFs.

Rule example:

```
tc filter add dev enp8s0f0_0 prot ip root flower dst_ip 7.7.7.7 action mirred egress redirect dev enp8s0f1
```

3.3.9.5.2 OVS-DPDK Hardware Offloads

3.3.9.5.2.1 OVS-DPDK Hardware Offloads Configuration

To configure OVS-DPDK HW offloads:

1. Unbind the VFs.

```
echo 0000:04:00.2 > /sys/bus/pci/drivers/mlx5_core/unbind
echo 0000:04:00.3 > /sys/bus/pci/drivers/mlx5_core/unbind
```

   Note: VMs with attached VFs must be powered off to be able to unbind the VFs.
2. Change the e-switch mode from Legacy to SwitchDev on the PF device (make sure all VFs are unbound). This will also create the VF representor netdevices in the host OS.

```
echo switchdev > /sys/class/net/enp4s0f0/compat/devlink/mode
```

   To revert to SR-IOV Legacy mode:

```
echo legacy > /sys/class/net/enp4s0f0/compat/devlink/mode
```

Note that running this command will also result in the removal of the VF representor netdevices.

3. Bind the VFs.

```
echo 0000:04:00.2 > /sys/bus/pci/drivers/mlx5_core/bind
echo 0000:04:00.3 > /sys/bus/pci/drivers/mlx5_core/bind
```

4. Run the Open vSwitch service.

```
systemctl start openvswitch
```

5. Enable hardware offload (disabled by default).

```
ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true
ovs-vsctl set Open_vSwitch . other_config:hw-offload=true
```

6. Configure the DPDK white list.

```
ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-extra="-a
0000:01:00.0,representor=[0],dv_flow_en=1,dv_esw_en=1,dv_xmeta_en=1"
```

⚠ Representor=[0-N]

7. Restart the Open vSwitch service. This step is required for HW offload changes to take effect.

```
systemctl restart openvswitch
```

8. Create OVS-DPDK bridge.

```
ovs-vsctl --no-wait add-br br0-ovs -- set bridge br0-ovs datapath_type=netdev
```

9. Add PF to OVS.

```
ovs-vsctl add-port br0-ovs pf -- set Interface pf type=dpdk options:dpdk-devargs=0000:88:00.0
```

10. Add representor to OVS.

```
ovs-vsctl add-port br0-ovs representor -- set Interface representor type=dpdk options:dpdk-
devargs=0000:88:00.0,representor=[0]
```

⚠ Representor=[0-N]

3.3.9.5.2.2 Offloading VXLAN Encapsulation/Decapsulation Actions

vSwitch in userspace rather than kernel-based Open vSwitch requires an additional bridge. The purpose of this bridge is to allow use of the kernel network stack for routing and ARP resolution.

The datapath needs to look-up the routing table and ARP table to prepare the tunnel header and transmit data to the output port.

The configuration is done with:
- PF on 0000:03:00.0 PCI and MAC 98:03:9b:cc:21:e8
- Local IP 56.56.67.1 - br-phy interface will be configured to this IP
- Remote IP 56.56.68.1

To configure OVS-DPDK VXLAN:

1. Create a br-phy bridge.

```
ovs-vsctl add-br br-phy -- set Bridge br-phy datapath_type=netdev -- br-set-external-id br-phy bridge-id
br-phy -- set bridge br-phy fail-mode=standalone other_config:hwaddr=98:03:9b:cc:21:e8
```

2. Attach PF interface to br-phy bridge.

```
ovs-vsctl add-port br-phy p0 -- set Interface p0 type=dpdk options:dpdk-devargs=0000:03:00.0
```

3. Configure IP to the bridge.

```
ip addr add 56.56.67.1/24 dev br-phy
```

4. Create a br-ovs bridge.

```
ovs-vsctl add-br br-ovs -- set Bridge br-ovs datapath_type=netdev -- br-set-external-id br-ovs bridge-id
br-ovs -- set bridge br-ovs fail-mode=standalone
```

5. Attach representor to br-ovs.

```
ovs-vsctl add-port br-ovs pf0vf0 -- set Interface pf0vf0 type=dpdk options:dpdk-
devargs=0000:03:00.0,representor=[0]
```

6. Add a port for the VXLAN tunnel.

```
ovs-vsctl add-port ovs-sriov vxlan0 -- set interface vxlan0 type=vxlan options:local_ip=56.56.67.1
 options:remote_ip=56.56.68.1 options:key=45 options:dst_port=4789
```

3.3.9.5.2.3 Connection Tracking Offload

Connection tracking enables stateful packet processing by keeping a record of currently open connections.
OVS flows using connection tracking can be accelerated using advanced Network Interface Cards (NICs) by offloading established connections.

To view offloaded connections, run:

```
ovs-appctl dpctl/offload-stats-show
```

3.3.9.5.2.4 SR-IOV VF LAG

To configure OVS-DPDK SR-IOV VF LAG:

1. Enable SR-IOV on the NICs.

```
mlxconfig -d <PCI> set SRIOV_EN=1
```

2. Allocate the desired number of VFs per port.

```
echo $n > /sys/class/net/<net name>/device/sriov_numvfs
```

3. Unbind all VFs.

```
echo <VF PCI> >/sys/bus/pci/drivers/mlx5_core/unbind
```

4. Change both NICs' mode to SwitchDev.

```
devlink dev eswitch set pci/<PCI> mode switchdev
```

5. Create Linux bonding using kernel modules.

```
modprobe bonding mode=<desired mode>
```

Note: Other bonding parameters can be added here. The supported Bond modes are: Active-Backup, XOR and LACP.

6. Bring all PFs and VFs down.

```
ip link set <PF/VF> down
```

7. Attach both PFs to the bond.

```
ip link set <PF> master bond0
```

8. To work with VF-LAG with OVS-DPDK, add the bond master (PF) to the bridge.

```
ovs-vsctl add-port br-phy p0 -- set Interface p0 type=dpdk options:dpdk-devargs=0000:03:00.0  options:dpdk-
lsc-interrupt=true
```

9. Add representor $N of PF0 or PF1 to a bridge.

```
ovs-vsctl add-port br-phy rep$N -- set Interface rep$N type=dpdk options:dpdk-devargs=<PF0
PCI>,representor=pf0vf$N

OR

ovs-vsctl add-port br-phy rep$N -- set Interface rep$N type=dpdk options:dpdk-devargs=<PF0
PCI>,representor=pf1vf$N
```

3.3.9.5.2.5 VirtIO Acceleration through VF Relay (Software & Hardware vDPA)

Hardware vDPA is supported on ConnectX-6 Dx, ConnectX-6 Lx & BlueField-2 cards and above only.

Hardware vDPA is enabled by default. In case your hardware does not support vDPA, the driver will fall back to Software vDPA.

To check which vDPA mode is activated on your driver, run: `ovs-ofctl -O OpenFlow14 dump-ports br0-ovs` and look for `hw-mode` flag.

⚠ This feature has not been accepted to the OVS-DPDK Upstream yet, making its API subject to change.

In user space, there are two main approaches for communicating with a guest (VM), either through SR-IOV, or through virtIO.

Phy ports (SR-IOV) allow working with port representor, which is attached to the OVS and a matching VF is given with pass-through to the guest. HW rules can process packets from up-link and direct them to the VF without going through SW (OVS). Therefore, using SR-IOV achieves the best performance.

However, SR-IOV architecture requires the guest to use a driver specific to the underlying HW. Specific HW driver has two main drawbacks:

1. Breaks virtualization in some sense (guest is aware of the HW). It can also limit the type of images supported.
2. Gives less natural support for live migration.

Using virtIO port solves both problems. However, it reduces performance and causes loss of some functionalities, such as, for some HW offloads, working directly with virtIO. To solve this conflict, a new netdev type- dpdkvdpa has been created. The new netdev is similar to the regular DPDK netdev, yet introduces several additional functionalities.

dpdkvdpa translates between phy port to virtIO port. It takes packets from the Rx queue and sends them to the suitable Tx queue, and allows transfer of packets from virtIO guest (VM) to a VF, and vice-versa, benefitting from both SR-IOV and virtIO.

To add vDPA port:

```
ovs-vsctl add-port br0 vdpa0 -- set Interface vdpa0 type=dpdkvdpa \
options:vdpa-socket-path=<sock path> \
options:vdpa-accelerator-devargs=<vf pci id> \
options:dpdk-devargs=<pf pci id>,representor=[id] \
options: vdpa-max-queues =<num queues> \
options: vdpa-sw=<true/false>
```

Note: vdpa-max-queues is an optional field. When the user wants to configure 32 vDPA ports, the maximum queues number is limited to 8.

vDPA Configuration in OVS-DPDK Mode

Prior to configuring vDPA in OVS-DPDK mode, follow the steps below.

1. Generate the VF.

```
echo 0 > /sys/class/net/enp175s0f0/device/sriov_numvfs
echo 4 > /sys/class/net/enp175s0f0/device/sriov_numvfs
```

2. Unbind each VF.

```
echo <pci> > /sys/bus/pci/drivers/mlx5_core/unbind
```

3. Switch to SwitchDev mode.

```
echo switchdev >> /sys/class/net/enp175s0f0/compat/devlink/mode
```

4. Bind each VF.

```
echo <pci> > /sys/bus/pci/drivers/mlx5_core/bind
```

5. Initialize OVS with:

```
ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true
ovs-vsctl --no-wait set Open_vSwitch . other_config:hw-offload=true
```

To configure vDPA in OVS-DPDK mode on ConnectX-5 cards and above:

1. Open vSwitch configuration.

```
ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-extra="-a
0000:01:00.0,representor=[0],dv_flow_en=1,dv_esw_en=1,dv_xmeta_en=1"
/usr/share/openvswitch/scripts/ovs-ctl restart
```

2. Create OVS-DPDK bridge.

```
ovs-vsctl add-br br0-ovs -- set bridge br0-ovs datapath_type=netdev
ovs-vsctl add-port br0-ovs pf -- set Interface pf type=dpdk options:dpdk-devargs=0000:01:00.0
```

3. Create vDPA port as part of the OVS-DPDK bridge.

```
ovs-vsctl add-port br0-ovs vdpa0 -- set Interface vdpa0 type=dpdkvdpa options:vdpa-socket-path=/var/run/
virtio-forwarder/sock0 options:vdpa-accelerator-devargs=0000:01:00.2 options:dpdk-
devargs=0000:01:00.0,representor=[0] options: vdpa-max-queues=8
```

To configure vDPA in OVS-DPDK mode on BlueField cards:

Set the bridge with the software or hardware vDPA port:

- On the ARM side:
  Create the OVS-DPDK bridge.

```
ovs-vsctl add-br br0-ovs -- set bridge br0-ovs datapath_type=netdev
ovs-vsctl add-port br0-ovs pf -- set Interface pf type=dpdk options:dpdk-devargs=0000:af:00.0
ovs-vsctl add-port br0-ovs rep-- set Interface rep type=dpdk options:dpdk-
devargs=0000:af:00.0,representor=[0]
```

- On the host side:
  Create the OVS-DPDK bridge.

```
ovs-vsctl add-br br1-ovs -- set bridge br1-ovs datapath_type=netdev protocols=OpenFlow14
ovs-vsctl add-port br0-ovs vdpa0 -- set Interface vdpa0 type=dpdkvdpa options:vdpa-socket-path=/var/run/
virtio-forwarder/sock0 options:vdpa-accelerator-devargs=0000:af:00.2
```

  Note: To configure SW vDPA, add "`options:vdpa-sw=true"` to the end of the command.

Software vDPA Configuration in OVS-Kernel Mode

SW vDPA can also be used in configurations where the HW offload is done through TC and not DPDK.

1. Open vSwitch configuration.

```
ovs-vsctl set Open_vSwitch . other_config:dpdk-extra="-a
0000:01:00.0,representor=[0],dv_flow_en=1,dv_esw_en=0,idv_xmeta_en=0,isolated_mode=1"
/usr/share/openvswitch/scripts/ovs-ctl restart
```

2. Create OVS-DPDK bridge.

```
ovs-vsctl add-br br0-ovs -- set bridge br0-ovs datapath_type=netdev
```

3. Create vDPA port as part of the OVS-DPDK bridge.

```
ovs-vsctl add-port br0-ovs vdpa0 -- set Interface vdpa0 type=dpdkvdpa options:vdpa-socket-path=/var/run/
virtio-forwarder/sock0 options:vdpa-accelerator-devargs=0000:01:00.2 options:dpdk-
devargs=0000:01:00.0,representor=[0] options: vdpa-max-queues=8
```

4. Create Kernel bridge.

```
ovs-vsctl add-br br-kernel
```

5. Add representors to Kernel bridge.

```
ovs-vsctl add-port br-kernel enp1s0f0_0
ovs-vsctl add-port br-kernel enp1s0f0
```

3.3.9.5.2.6 Large MTU/Jumbo Frame Configuration

To configure MTU/jumbo frames:

1. Verify that the Kernel version on the VM is 4.14 or above.

```
cat /etc/redhat-release
```

2. Set the MTU on both physical interfaces in the host.

```
ifconfig ens4f0 mtu 9216
```

3. Send a large size packet and verify that it is sent and received correctly.

```
tcpdump -i ens4f0 -nev icmp &
ping 11.100.126.1 -s 9188 -M do -c 1
```

4. Enable host_mtu in xml, and add the following values to xml.

```
host_mtu=9216,csum=on,guest_csum=on,host_tso4=on,host_tso6=on
```

   Example:

```
<qemu:commandline>
<qemu:arg value='-chardev'/>
<qemu:arg value='socket,id=charnet1,path=/tmp/sock0,server'/>
<qemu:arg value='-netdev'/>
<qemu:arg value='vhost-user,chardev=charnet1,queues=16,id=hostnet1'/>
<qemu:arg value='-device'/>
<qemu:arg value='virtio-net-
pci,mq=on,vectors=34,netdev=hostnet1,id=net1,mac=00:21:21:24:02:01,bus=pci.0,addr=0xC,page-per-
vq=on,rx_queue_size=1024,tx_queue_size=1024,host_mtu=9216,csum=on,guest_csum=on,host_tso4=on,host_tso6=on'/
>
</qemu:commandline>
```

5. Add mtu_request=9216 option to the OvS ports inside the container and restart the OVS:

```
ovs-vsctl add-port br0-ovs pf -- set Interface pf type=dpdk options:dpdk-devargs=0000:c4:00.0
 mtu_request=9216
```

OR:

```
ovs-vsctl add-port br0-ovs vdpa0 -- set Interface vdpa0 type=dpdkvdpa options:vdpa-socket-path=/tmp/sock0
options:vdpa-accelerator-devargs=0000:c4:00.2 options:dpdk-devargs=0000:c4:00.0,representor=[0]
mtu_request=9216
/usr/share/openvswitch/scripts/ovs-ctl restart
```

6. Start the VM and configure the MTU on the VM.

```
ifconfig eth0 11.100.124.2/16 up
ifconfig eth0 mtu 9216
ping 11.100.126.1 -s 9188 -M do -c1
```

3.3.9.5.2.7 E2E Cache

This feature is at beta level.

OVS offload rules are based on a multi-table architecture. E2E cache feature enables merging the multi-table flow matches and actions into one joint flow.

This improves connection tracking performance by using a single-table when exact match is detected.

- *To set the E2E cache size (default = 4k):*

```
ovs-vsctl set open_vswitch . other_config:e2e-size=<size>
systemctl restart openvswitch
```

Note: Make sure to restart the openvswitch service in order for the configuration to take effect.

- *To enable/disable E2E cache (default = disabled) :*

```
ovs-vsctl set open_vswitch . other_config:e2e-enable=<true/false>
systemctl restart openvswitch
```

Note: Make sure to restart the openvswitch service in order for the configuration to take effect.

- *To run E2E cache statistics:*

```
ovs-appctl dpctl/dump-e2e-stats
```

- *To run E2E cache flows:*

```
ovs-appctl dpctl/dump-e2e-flows
```

3.3.9.5.2.8 Geneve Encapsulation/Decapsulation

Geneve tunneling offload feature support includes matching on extension header.

To configure OVS-DPDK Geneve encap/decap:

1. Create a br-phy bridge.

```
ovs-vsctl --may-exist add-br br-phy -- set Bridge br-phy datapath_type=netdev -- br-set-external-id br-phy
bridge-id br-phy -- set bridge br-phy fail-mode=standalone
```

2. Attach PF interface to br-phy bridge.

```
ovs-vsctl add-port br-phy pf -- set Interface pf type=dpdk options:dpdk-devargs=<PF PCI>
```

3. Configure IP to the bridge.

```
ifconfig br-phy <$local_ip_1> up
```

4. Create a br-int bridge.

```
ovs-vsctl --may-exist add-br br-int -- set Bridge br-int datapath_type=netdev -- br-set-external-id br-int
 bridge-id br-int -- set bridge br-int fail-mode=standalone
```

5. Attach representor to br-int.

```
ovs-vsctl add-port br-int rep$x -- set Interface rep$x type=dpdk options:dpdk-devargs=<PF
PCI>,representor=[$x]
```

6. Add a port for the GENEVE tunnel.

```
ovs-vsctl add-port br-int geneve0 -- set interface geneve0 type=geneve options:key=<VNI>
options:remote_ip=<$remote_ip_1> options:local_ip=<$local_ip_1>
```

3.3.9.5.2.9 Parallel Offloads

OVS-DPDK supports parallel insertion and deletion of offloads (flow & CT). While multiple threads are supported, by default only one is used.

To configure multiple threads:

```
ovs-vsctl set Open_vSwitch . other_config:n-offload-threads=3
```

Make sure to restart the openvswitch service in order for the configuration to take effect.

```
systemctl restart openvswitch
```

For more information, see the OvS user manual.

3.3.9.5.2.10 sFlow

This feature allows for monitoring traffic sent between two VMs on the same host using an sFlow collector.

To sample all traffic over the OVS bridge, run the following:

```
# ovs-vsctl -- --id=@sflow create sflow agent=\"$SFLOW_AGENT\" \
             target=\"$SFLOW_TARGET:$SFLOW_HEADER\" header=$SFLOW_HEADER \
             sampling=$SFLOW_SAMPLING polling=10 \
             -- set bridge  sflow=@sflow
```

Parameter	Description
SFLOW_AGENT	Indicates that the sFlow agent should send traffic from SFLOW_AGENT's IP address
SFLOW_TARGET	Remote IP address of the sFLOW collector
SFLOW_PORT	Remote IP destination port of the sFlow collector
SFLOW_HEADER	Size of packet header to sample (in bytes)
SFLOW_SAMPLING	Sample rate

To clear the sFLOW configuration, runt he following:

```
# ovs-vsctl clear bridge br-vxlan mirrors
```

Currently sFlow for OVS-DPDK is supported without CT.

3.3.9.5.2.11 CT CT NAT

To enable ct-ct-nat offloads in OvS-DPDK, execute the following command (deafult value is false):

```
ovs-vsctl set open_vswitch . other_config:ct-action-on-nat-conns=true
```

If disabled, ct-ct-nat configurations will not be fully offloaded, improving connection offloading rate for
other cases (ct and ct-nat).

If enabled, ct-ct-nat configurations will be fully offloaded but ct and ct-nat offloading
will be slower to be created.

3.3.9.5.2.12 OpenFlow Meters (OpenFlow13+):

OpenFlow meters in OVS are implemented according to RFC 2697 (Single Rate Three Color Marker—srTCM).

- The srTCM meters an IP packet stream and marks its packets either green, yellow, or red. The color is decided on a Committed Information Rate (CIR) and two associated burst sizes, Committed Burst Size (CBS), and Excess Burst Size (EBS).
- A packet is marked green if it does not exceed the CBS, yellow if it exceeds the CBS but not the EBS, and red otherwise.
- The volume of green packets should never be smaller than the CIR.

To configure a meter in OVS:

1. Create a meter over a certain bridge:

   a.
   ```
   ovs-ofctl -O openflow13 add-meter $bridge
   meter=$id,$pktps/$kbps,band=type=drop,rate=$rate,[burst,burst_size=$burst_size]
   ```

   b. Parameters:

Parameter	Description
bridge	Name of the bridge on which the meter will be applied.
id	Unique meter ID (32 bits) which will be used as an identifier for the meter.
pktps/kbps	Indication if the meter should work according to packets-per-second or kilobits-per-second.
rate	Rate of pktps/kbps of allowed data transmission.
burst	If set, enables burst support for meter bands through the “burst_size” parameter.
burst_size	If burst is specified for the meter entry, configures the maximum burst allowed for the band in kilobits/packets, depending on whether kbps or pktps was specified. If unspecified, the switch is free to select some reasonable value depending on its configuration. Currently, if burst was not specified, the burst_size parameter is set as the “rate”.

2. Add the meter to a certain OpenFlow rule. For example:

```
ovs-ofctl -O openflow13 add-flow $bridge "table=0,actions=meter:$id,normal“
```

3. View the meter statistics:

```
ovs-ofctl -O openflow13 meter-stats $bridge meter=$id
```

4. For more information, refer to openvswitch documentation http://www.openvswitch.org/support/dist-docs/ovs-ofctl.8.txt

3.3.9.6 VirtIO Acceleration through Hardware vDPA

3.3.9.6.1 Hardware vDPA Installation

Hardware vDPA requires QEMU v2.12 (or with upstream 6.1.0) and DPDK v20.11 as minimal versions.

To install QEMU:

1. Clone the sources:

```
git clone https://git.qemu.org/git/qemu.git
cd qemu
git checkout v2.12
```

2. Build QEMU:

```
mkdir bin
cd bin
../configure --target-list=x86_64-softmmu --enable-kvm
make -j24
```

To install DPDK:

1. Clone the sources:

```
git clone git://dpdk.org/dpdk
cd dpdk
```

```
git checkout v20.11
```

2. Install dependencies (if needed):

```
yum install cmake gcc libnl3-devel libudev-devel make pkgconfig valgrind-devel pandoc libibverbs libmlx5
libmnl-devel -y
```

3. Configure DPDK:

```
export RTE_SDK=$PWD
make config T=x86_64-native-linuxapp-gcc
cd build
sed -i 's/\(CONFIG_RTE_LIBRTE_MLX5_PMD=\)n/\1y/g' .config
sed -i 's/\(CONFIG_RTE_LIBRTE_MLX5_VDPA_PMD=\)n/\1y/g' .config
```

4. Build DPDK:

```
make -j
```

5. Build the vDPA application:

```
cd $RTE_SDK/examples/vdpa/
make -j
```

3.3.9.6.2 Hardware vDPA Configuration

To configure huge pages:

```
mkdir -p /hugepages
mount -t hugetlbfs hugetlbfs /hugepages
echo <more> > /sys/devices/system/node/node0/hugepages/hugepages-1048576kB/nr_hugepages
echo <more> > /sys/devices/system/node/node1/hugepages/hugepages-1048576kB/nr_hugepages
```

To configure a vDPA VirtIO interface in an existing VM's xml file (using libvirt):

1. Open the VM's configuration xml for editing:

```
virsh edit <domain name>
```

2. Modify/add the following:
    a. Change the top line to:

```
<domain type='kvm' xmlns:qemu='http://libvirt.org/schemas/domain/qemu/1.0'>
```

    b. Assign a memory amount and use 1GB page size for hugepages (size must be the same as used for the vDPA application), so that the memory configuration looks like the following.

```
<memory unit='KiB'>4194304</memory>
<currentMemory unit='KiB'>4194304</currentMemory>
<memoryBacking>
  <hugepages>
    <page size='1048576' unit='KiB'/>
  </hugepages>
</memoryBacking>
```

    c. Assign an amount of CPUs for the VM CPU configuration, so that the `vcpu` and `cputune` configuration looks like the following.

```
<vcpu placement='static'>5</vcpu>
<cputune>
```

```
  <vcpupin vcpu='0' cpuset='14'/>
  <vcpupin vcpu='1' cpuset='16'/>
  <vcpupin vcpu='2' cpuset='18'/>
  <vcpupin vcpu='3' cpuset='20'/>
  <vcpupin vcpu='4' cpuset='22'/>
</cputune>
```

d. Set the memory access for the CPUs to be shared, so that the `cpu` configuration looks like the following.

```
<cpu mode='custom' match='exact' check='partial'>
  <model fallback='allow'>Skylake-Server-IBRS</model>
  <numa>
    <cell id='0' cpus='0-4' memory='8388608' unit='KiB' memAccess='shared'/>
  </numa>
</cpu>
```

e. Set the emulator in use to be the one built in step "2. Build QEMU" above, so that the emulator configuration looks as follows.

```
<emulator><path to qemu executable></emulator>
```

f. Add a virtio interface using qemu command line argument entries, so that the new interface snippet looks as follows.

```
<qemu:commandline>
  <qemu:arg value='-chardev'/>
  <qemu:arg value='socket,id=charnet1,path=/tmp/sock-virtio0'/>
  <qemu:arg value='-netdev'/>
  <qemu:arg value='vhost-user,chardev=charnet1,queues=16,id=hostnet1'/>
  <qemu:arg value='-device'/>
  <qemu:arg value='virtio-net-
pci,mq=on,vectors=6,netdev=hostnet1,id=net1,mac=e4:11:c6:d3:45:f2,bus=pci.0,addr=0x6,
   page-per-vq=on,rx_queue_size=1024,tx_queue_size=1024'/>
</qemu:commandline>
```

Note: In this snippet, the vhostuser socket file path, the amount of queues, the MAC and the PCI slot of the VirtIO device can be configured.

3.3.9.6.3 Running Hardware vDPA

⚠ Hardware vDPA supports SwitchDev mode only.

Create the ASAP² environment:

1. Create the VFs.
2. Enter switchdev mode.
3. Set up OVS.

***Run** the vDPA application.*

```
cd $RTE_SDK/examples/vdpa/build
./vdpa -w <VF PCI BDF>,class=vdpa --log-level=pmd,info -- -i
```

Create a vDPA port via the vDPA application CLI.

```
create /tmp/sock-virtio0 <PCI DEVICE BDF>
```

Note: The vhostuser socket file path must be the one used when configuring the VM.

Start the VM.

```
virsh start <domain name>
```

For further information on the vDPA application, please visit: https://doc.dpdk.org/guides/sample_app_ug/vdpa.html.

3.3.9.7 Bridge Offload

- Bridge offload is supported on ConnectX-6 Dx NIC
- Bridge offload is supported SwitchDev mode only
- Bridge offload is supported from kernel version 5.15

A Linux bridge is in-kernel software network switch (based on and implements subset of IEEE 802.1D standard) that is used to connect Ethernet segments together in a protocol-independent way. Packets are forwarded based on L2 Ethernet header addresses.

mlx5 provides capabilities to offload bridge data-plane unicast packet forwarding and VLAN management to hardware.

3.3.9.7.1 Basic Configuration

1. Initialize the ASAP^2 environment:
   a. Create the VFs.
   b. Enter switchdev mode.
2. Create a bridge and add mlx5 representors to bridge:

```
ip link add name bridge0 type bridge
ip link set enp8s0f0_0 master bridge0
```

3.3.9.7.2 Configuring VLAN

1. Enable VLAN filtering on the bridge.

```
ip link set bridge0 type bridge vlan_filtering 1
```

2. Configure port VLAN matching (trunk mode). In this configuration, only packets with specified VID are allowed.

```
bridge vlan add dev enp8s0f0_0 vid 2
```

3. Configure port VLAN tagging (access mode). In this configuration VLAN header in pushed/popped on reception/transmission on port.

```
bridge vlan add dev enp8s0f0_0 vid 2 pvid untagged
```

3.3.9.7.3 VF LAG Support

Bridge supports offloading on bond net device that is fully initialized with mlx5 uplink representors and is in single (shared) FDB LAG mode. Details about initialization of LAG are provided in SR-IOV VF LAG section, above.

Add bonding net device to bridge.

```
ip link set bond0 master bridge0
```

For further information on interacting with Linux bridge via iproute2 bridge tool, please consult man page (man 8 bridge).

3.3.9.8 Appendix: NVIDIA Firmware Tools

Download and install the MFT package corresponding to your computer's operating system. You would need the kernel-devel or kernel-headers RPM before the tools are built and installed.
The package is available at nvidia.com/en-us/networking/ → Products → Software → Firmware Tools.

1. Start the mst driver.

```
# mst start
Starting MST (Mellanox Software Tools) driver set
Loading MST PCI module - Success
Loading MST PCI configuration module - Success
Create devices
```

2. Show the devices status.

```
ST modules:
------------
    MST PCI module loaded
    MST PCI configuration module loaded

PCI devices:
------------
DEVICE_TYPE             MST                           PCI     RDMA NET        NUMA
ConnectX4lx(rev:0)        /dev/mst/mt4117_pciconf0.1    04:00.1        net-enp4s0f1  NA
ConnectX4lx(rev:0)        /dev/mst/mt4117_pciconf0      04:00.0        net-enp4s0f0  NA


# mlxconfig -d /dev/mst/mt4117_pciconf0 q | head -16

Device #1:
----------

Device type:    ConnectX4lx
PCI device:     /dev/mst/mt4117_pciconf0

Configurations:                                 Current
         SRIOV_EN                               True(1)
         NUM_OF_VFS                             8
         PF_LOG_BAR_SIZE                        5
         VF_LOG_BAR_SIZE                        5
         NUM_PF_MSIX                            63
         NUM_VF_MSIX                            11
         LINK_TYPE_P1                           ETH(2)
         LINK_TYPE_P2                           ETH(2)
```

3. Make sure your configuration is as follows:
   * SR-IOV is enabled (SRIOV_EN=1)
   * The number of enabled VFs is enough for your environment (NUM_OF_VFS=N)
   * The port's link type is Ethernet (LINK_TYPE_P1/2=2) when applicable

   If this is not the case, use mlxconfig to enable that, as follows:
   a. Enable SR-IOV.

```
# mlxconfig -d /dev/mst/mt4115_pciconf0 s SRIOV_EN=1
```

b. Set the number of required VFs.

```
# mlxconfig -d /dev/mst/mt4115_pciconf0 s NUM_OF_VFS=8
```

c. Set the link type to Ethernet.

```
# mlxconfig -d /dev/mst/mt4115_pciconf0 s LINK_TYPE_P1=2
# mlxconfig -d /dev/mst/mt4115_pciconf0 s LINK_TYPE_P2=2
```

4. Conduct a cold reboot (or a firmware reset).

```
# mlxfwreset -d /dev/mst/mt4115_pciconf0 reset
```

5. Query the firmware to make sure everything is set correctly.

```
# mlxconfig -d /dev/mst/mt4115_pciconf0 q
```

3.4 Programming

This chapter is aimed for application developers and expert users that wish to develop applications over MLNX_OFED.

3.4.1 Raw Ethernet Programming

Raw Ethernet programming enables writing an application that bypasses the kernel stack. To achieve this, packet headers and offload options need to be provided by the application.
For a basic example on how to use Raw Ethernet programming, refer to the Raw Ethernet Programming: Basic Introduction—Code Example Community post.

3.4.1.1 Packet Pacing

Packet pacing is a raw Ethernet sender feature that enables controlling the rate of each QP, per send queue.
For a basic example on how to use packet pacing per flow over libibverbs, refer to Raw Ethernet Programming: Packet Pacing—Code Example Community post.

3.4.1.2 TCP Segmentation Offload (TSO)

TCP Segmentation Offload (TSO) enables the adapter cards to accept a large amount of data with a size greater than the MTU size. The TSO engine splits the data into separate packets and inserts the user-specified L2/L3/L4 headers automatically per packet. With the usage of TSO, CPU is offloaded from dealing with a large throughput of data.
To be able to program that on the sender side, refer to the Raw Ethernet Programming: TSO—Code Example Community post.

3.4.1.3 ToS Based Steering

ToS/DSCP is an 8-bit field in the IP packet that enables different service levels to be assigned to network traffic. This is achieved by marking each packet in the network with a DSCP code and appropriating the corresponding level of service to it.
To be able to steer packets according to the ToS field on the receiver side, refer to the Raw Ethernet Programming: ToS—Code Example Community post.

3.4.1.4 Flow ID Based Steering

Flow ID based steering enables developing a code that will steer packets using flow ID when developing Raw Ethernet over verbs. For more information on flow ID based steering, refer to the Raw Ethernet Programming: Flow ID Steering—Code Example Community post.

3.4.1.5 VXLAN Based Steering

VXLAN based steering enables developing a code that will steer packets using the VXLAN tunnel ID when developing Raw Ethernet over verbs. For more information on VXLAN based steering, refer to the Raw Ethernet Programming: VXLAN Steering—Code Example Community post.

3.4.2 Device Memory Programming

> ⚠ This feature is supported on ConnectX-5/ConnectX-5 Ex adapter cards and above only.

Device Memory is an API that allows using on-chip memory located on the device as a data buffer for send/receive and RDMA operations. The device memory can be mapped and accessed directly by user and kernel applications, and can be allocated in various sizes, registered as memory regions with local and remote access keys for performing the send/receive and RDMA operations.
Using the device memory to store packets for transmission can significantly reduce transmission latency compared to the host memory.

3.4.2.1 Device Memory Programming Model

The new API introduces a similar procedure to the host memory for sending packets from the buffer:

- ibv_alloc_dm()/ibv_free_dm() - to allocate/free device memory
- ibv_reg_dm_mr - to register the allocated device memory buffer as a memory region and get a memory key for local/remote access by the device
- ibv_memcpy_to_dm - to copy data to a device memory buffer
- ibv_memcpy_from_dm - to copy data from a device memory buffer
- ibv_post_send/ibv_post_receive - to request the device to perform a send/receive operation using the memory key

For examples, see Device Memory.

3.4.3 RDMA-CM QP Timeout Control

RDMA-CM QP Timeout Control feature enables users to control the QP timeout for QPs created with RDMA-CM.

A new option in 'rdma_set_option' function has been added to enable overriding calculated QP timeout, in order to provide QP attributes for QP modification. To achieve that, rdma_set_option() should be called with the new flag `RDMA_OPTION_ID_ACK_TIMEOUT`. Example:

```
rdma_set_option(cma_id, RDMA_OPTION_ID, RDMA_OPTION_ID_ACK_TIMEOUT, &timeout, sizeof(timeout));
```

3.4.4 RDMA-CM Application Managed QP

Applications which do not create a QP through rdma_create_qp() may want to postpone the ESTABLISHED event on the passive side, to let the active side complete an application-specific connection establishment phase. For example, modifying the init state of the QP created by the application to RTR state, or make some preparations for receiving messages from the passive side. The feature returns a new event on the active side: CONNECT_RESPONSE, instead of ESTABLISHED, if id->qp==NULL. This gives the application a chance to perform the extra connection setup. Afterwards, the new rdma_establish() API should be called to complete the connection and generate an ESTABLISHED event on the passive side.

In addition, this feature exposes the 'rdma_init_qp_attr' function in librdmacm API, which enables applications to get the parameters for creating Address Handler (AH) or control QP attributes after its creation.

3.5 InfiniBand Fabric Utilities

This section first describes common configuration, interface, and addressing for all the tools in the package.

3.5.1 Common Configuration, Interface and Addressing

3.5.1.1 Topology File (Optional)

An InfiniBand fabric is composed of switches and channel adapter (HCA/TCA) devices. To identify devices in a fabric (or even in one switch system), each device is given a GUID (a MAC equivalent). Since a GUID is a non-user-friendly string of characters, it is better to alias it to a meaningful, user-given name. For this objective, the IB Diagnostic Tools can be provided with a “topology file”, which is an optional configuration file specifying the IB fabric topology in user-given names.
For diagnostic tools to fully support the topology file, the user may need to provide the local system name (if the local hostname is not used in the topology file).
To specify a topology file to a diagnostic tool use one of the following two options:

1. On the command line, specify the file name using the option ‘-t <topology file name>’
2. Define the environment variable IBDIAG_TOPO_FILE

To specify the local system name to an diagnostic tool use one of the following two options:

1. On the command line, specify the system name using the option '-s <local system name>'
2. Define the environment variable IBDIAG_SYS_NAME

3.5.2 InfiniBand Interface Definition

The diagnostic tools installed on a machine connect to the IB fabric by means of an HCA port through which they send MADs. To specify this port to an IB diagnostic tool use one of the following options:

1. On the command line, specify the port number using the option '-p <local port number>' (see below)
2. Define the environment variable IBDIAG_PORT_NUM

In case more than one HCA device is installed on the local machine, it is necessary to specify the device's index to the tool as well. For this use on of the following options:

1. On the command line, specify the index of the local device using the following option: '-i <index of local device>'
2. Define the environment variable IBDIAG_DEV_IDX

3.5.3 Addressing

This section applies to the ibdiagpath tool only. A tool command may require defining the destination device or port to which it applies.

The following addressing modes can be used to define the IB ports:

- Using a Directed Route to the destination: (Tool option '-d')
  This option defines a directed route of output port numbers from the local port to the destination.
- Using port LIDs: (Tool option '-l'):
  In this mode, the source and destination ports are defined by means of their LIDs. If the fabric is configured to allow multiple LIDs per port, then using any of them is valid for defining a port.
- Using port names defined in the topology file: (Tool option '-n')
  This option refers to the source and destination ports by the names defined in the topology file. (Therefore, this option is relevant only if a topology file is specified to the tool.) In this mode, the tool uses the names to extract the port LIDs from the matched topology, then the tool operates as in the '-l' option.

3.5.4 Diagnostic Utilities

The diagnostic utilities described in this chapter provide means for debugging the connectivity and status of InfiniBand (IB) devices in a fabric.

Diagnostic Utilities

Utility	Description
dump_fts	Dumps tables for every switch found in an ibnetdiscover scan of the subnet. The dump file format is compatible with loading into OpenSM using the -R file -U /path/to/dump-file syntax. For further information, please refer to the tool's man page.
ibaddr	Can be used to show the LID and GID addresses of the specified port or the local port by default. This utility can be used as simple address resolver. For further information, please refer to the tool's man page.
ibcacheedit	Allows users to edit an ibnetdiscover cache created through the --cache option in ibnetdiscover(8). For further information, please refer to the tool's man page.
ibcccconfig	Supports the configuration of congestion control settings on switches and HCAs. For further information, please refer to the tool's man page.
ibcccquery	Supports the querying of settings and other information related to congestion control. For further information, please refer to the tool's man page.
ibcongest	Provides static congestion analysis. It calculates routing for a given topology (topo-mode) or uses extracted lst/fdb files (lst-mode). Additionally, it analyzes congestion for a traffic schedule provided in a "schedule-file" or uses an automatically generated schedule of all-to-all-shift. To display a help message which details the tool's options, please run "/opt/ ibutils2/bin/ibcongest -h". For further information, please refer to the tool's man page.
ibdev2netdev	Enables association between IB devices and ports and the associated net device. Additionally it reports the state of the net device link. For further information, please refer to the tool's man page.
ibdiagnet (of ibutils2)	Scans the fabric using directed route packets and extracts all the available information regarding its connectivity and devices. An ibdiagnet run performs the following stages: • Fabric discovery • Duplicated GUIDs detection • Links in INIT state and unresponsive links detection • Counters fetch • Error counters check • Routing checks • Link width and speed checks • Alias GUIDs check • Subnet Manager check • Partition keys check • Nodes information **Note:** This version of ibdiagnet is included in the ibutils2 package, and it is run by default after installing NVIDIA OFED. To use this ibdiagnet version, run: ibdiagnet. For further information, either: 1. Run `ibdiagnet -H` Or 2. Refer to docs.nvidia.com/networking/display/ibdiagnetUserManualv10

Utility	Description
ibdiagpath	Traces a path between two end-points and provides information regarding the nodes and ports traversed along the path. It utilizes device specific health queries for the different devices along the path. The way ibdiagpath operates depends on the addressing mode used in the command line. If directed route addressing is used (--dr_path flag), the local node is the source node and the route to the destination port is known apriori (for example: ibdiagpath --dr_path 0,1). On the other hand, if LID-route addressing is employed, --src_lid and --dest_lid, then the source and destination ports of a route are specified by their LIDs. In this case, the actual path from the local port to the source port, and from the source port to the destination port, is defined by means of Subnet Management Linear Forwarding Table queries of the switch nodes along that path. Therefore, the path cannot be predicted as it may change. Example: ibdiagpath --src_lid 1 --dest_lid 28 For further information, please refer to the tool's -help flag.
ibdump	Dump InfiniBand traffic that flows to and from NVIDIA's ConnectX® family adapters InfiniBand ports. Note the following: • ibdump is not supported for Virtual functions (SR-IOV) • Infiniband traffic sniffing is supported on all HCAs The dump file can be loaded by the Wireshark tool for graphical traffic analysis. The following describes a workflow for local HCA (adapter) sniffing: 1. Run ibdump with the desired options 2. Run the application that you wish its traffic to be analyzed 3. Stop ibdump (CTRL-C) or wait for the data buffer to fill (in --mem-mode) 4. Open Wireshark and load the generated file To download Wireshark for a Linux or Windows environment go to www.wireshark.org. **Notes:** • Although ibdump is a Linux application, the generated .pcap file may be analyzed on either operating system. • If one of the HCA's ports is configured as InfiniBand, ibdump requires IPoIB DMFS to be enabled. For further information, please refer to Flow Steering Configuration section. For further information, please refer to the tool's man page.
iblinkinfo	Reports link info for each port in an InfiniBand fabric, node by node. Optionally, iblinkinfo can do partial scans and limit its output to parts of a fabric. For further information, please refer to the tool's man page.
ibnetdiscover	Performs InfiniBand subnet discovery and outputs a human readable topology file. GUIDs, node types, and port numbers are displayed as well as port LIDs and node descriptions. All nodes (and links) are displayed (full topology). This utility can also be used to list the current connected nodes. The output is printed to the standard output unless a topology file is specified. For further information, please refer to the tool's man page.
ibnetsplit	Automatically groups hosts and creates scripts that can be run in order to split the network into sub-networks containing one group of hosts. For further information, please refer to the tool's man page.
ibnodes	Uses the current InfiniBand subnet topology or an already saved topology file and extracts the InfiniBand nodes (CAs and switches). For further information, please refer to the tool's man page.
ibping	Uses vendor mads to validate connectivity between InfiniBand nodes. On exit, (IP) ping like output is show. ibping is run as client/server. The default is to run as client. Note also that a default ping server is implemented within the kernel. For further information, please refer to the tool's man page.

Utility	Description
ibportstate	Enables querying the logical (link) and physical port states of an InfiniBand port. It also allows adjusting the link speed that is enabled on any InfiniBand port. If the queried port is a switch port, then ibportstate can be used to: • disable, enable or reset the port • validate the port's link width and speed against the peer port In case of multiple channel adapters (CAs) or multiple ports without a CA/ port being specified, a port is chosen by the utility according to the following criteria: • The first ACTIVE port that is found. • If not found, the first port that is UP (physical link state is LinkUp). For further information, please refer to the tool's man page.
ibqueryerrors	The default behavior is to report the port error counters which exceed a threshold for each port in the fabric. The default threshold is zero (0). Error fields can also be suppressed entirely. In addition to reporting errors on every port, ibqueryerrors can report the port transmit and receive data as well as report full link information to the remote port if available. For further information, please refer to the tool's man page.
ibroute	Uses SMPs to display the forwarding tables—unicast (LinearForwarding- Table or LFT) or multicast (MulticastForwardingTable or MFT)—for the specified switch LID and the optional lid (mlid) range. The default range is all valid entries in the range 1 to FDBTop. For further information, please refer to the tool's man page.
ibstat	ibstat is a binary which displays basic information obtained from the local IB driver. Output includes LID, SMLID, port state, link width active, and port physical state. For further information, please refer to the tool's man page.
ibstatus	Displays basic information obtained from the local InfiniBand driver. Output includes LID, SMLID, port state, port physical state, port width and port rate. For further information, please refer to the tool's man page.
ibswitches	Traces the InfiniBand subnet topology or uses an already saved topology file to extract the InfiniBand switches. For further information, please refer to the tool's man page.
ibsysstat	Uses vendor mads to validate connectivity between InfiniBand nodes and obtain other information about the InfiniBand node. ibsysstat is run as client/ server. The default is to run as client. For further information, please refer to the tool's man page.
ibtopodiff	Compares a topology file and a discovered listing ofsubnet.lst/ibdiagnet.lst and reports mismatches. Two different algorithms provided: • Using the -e option is more suitable for MANY mismatches it applies less heuristics and provide details about the match • Providing the -s, -p and -g starts a detailed heuristics that should be used when only small number of changes are expected For further information, please refer to the tool's man page.
ibtracert	Uses SMPs to trace the path from a source GID/LID to a destination GID/ LID. Each hop along the path is displayed until the destination is reached or a hop does not respond. By using the -m option, multicast path tracing can be performed between source and destination nodes. For further information, please refer to the tool's man page.
ibv_asyncwatch	Display asynchronous events forwarded to userspace for an InfiniBand device. For further information, please refer to the tool's man page.
ibv_devices	Lists InfiniBand devices available for use from userspace, including node GUIDs. For further information, please refer to the tool's man page.

Utility	Description
ibv_devinfo	Queries InfiniBand devices and prints about them information that is available for use from userspace. For further information, please refer to the tool's man page.
mstflint	Queries and burns a binary firmware-image file on non-volatile (Flash) memories of NVIDIA InfiniBand and Ethernet network adapters. The tool requires root privileges for Flash access. To run mstflint, you must know the device location on the PCI bus. **Note:** If you purchased a standard NVIDIA network adapter card, please download the firmware image from nvidia.com/en-us/networking/ → Support → Support → Firmware Download. If you purchased a non-standard card from a vendor other than NVIDIA, please contact your vendor. For further information, please refer to the tool's man page.
perfquery	Queries InfiniBand ports' performance and error counters. Optionally, it displays aggregated counters for all ports of a node. It can also reset counters after reading them or simply reset them. For further information, please refer to the tool's man page.
saquery	Issues the selected SA query. Node records are queried by default. For further information, please refer to the tool's man page.
sminfo	Issues and dumps the output of an sminfo query in human readable format. The target SM is the one listed in the local port info or the SM specified by the optional SM LID or by the SM direct routed path. **Note:** Using sminfo for any purpose other than a simple query might result in a malfunction of the target SM. For further information, please refer to the tool's man page.
smparquery	Sends SMP query for adaptive routing and private LFT features. For further information, please refer to the tool's man page.
smpdump	A general purpose SMP utility which gets SM attributes from a specified SMA. The result is dumped in hex by default. For further information, please refer to the tool's man page.
smpquery	Provides a basic subset of standard SMP queries to query Subnet management attributes such as node info, node description, switch info, and port info. For further information, please refer to the tool's man page.

3.5.4.1 Link Level Retransmission (LLR) in FDR Links

With the introduction of FDR 56 Gbps technology, NVIDIA enabled a proprietary technology called LLR (Link Level Retransmission) to improve the reliability of FDR links.

This proprietary LLR technology adds additional CRC checking to the data stream and retransmits portions of packets with CRC errors at the local link level. Customers should be aware of the following facts associated with LLR technology:

- Traditional methods of checking the link health can be masked because the LLR technology automatically fixes errors. The traditional IB symbol error counter will show no errors when LLR is active.
- Latency of the fabric can be impacted slightly due to LLR retransmissions. Traditional IB performance utilities can be used to monitor any latency impact.
- Bandwidth of links can be reduced if cable performance degrades and LLR retransmissions become too numerous. Traditional IB bandwidth performance utilities can be used to monitor any bandwidth impact.

Due to these factors, an LLR retransmission rate counter has been added to the ibdiagnet utility that can give end users an indication of the link health.

To monitor LLR retransmission rate:

1. Run ibdiagnet, no special flags required.
2. If the LLR retransmission rate limit is exceeded it will print to the screen.
3. The default limit is set to 500 and requires further investigation if exceeded.
4. The LLR retransmission rate is reflected in the results file /var/tmp/ibdiagnet2/ ibdiagnet2.pm.

The default value of 500 retransmissions/sec has been determined by NVIDIA based on the extensive simulations and testing. Links exhibiting a lower LLR retransmission rate should not raise special concern.

3.5.5 Performance Utilities

The performance utilities described in this chapter are intended to be used as a performance micro-benchmark.

Utility	Description
ib_atomic_bw	Calculates the BW of RDMA Atomic transactions between a pair of machines. One acts as a server and the other as a client. The client RDMA sends atomic operation to the server and calculate the BW by sampling the CPU each time it receive a successful completion. The test supports features such as Bidirectional, in which they both RDMA atomic to each other at the same time, change of MTU size, tx size, number of iteration, message size and more. Using the "-a" flag provides results for all message sizes. For further information, please refer to the tool's man page.
ib_atomic_lat	Calculates the latency of RDMA Atomic transaction of message_size between a pair of machines. One acts as a server and the other as a client. The client sends RDMA atomic operation and sample the CPU clock when it receives a successful completion, in order to calculate latency. For further information, please refer to the tool's man page.
ib_read_bw	Calculates the BW of RDMA read between a pair of machines. One acts as a server and the other as a client. The client RDMA reads the server memory and calculate the BW by sampling the CPU each time it receive a successful completion. The test supports features such as Bidirectional, in which they both RDMA read from each other memory's at the same time, change of MTU size, tx size, number of iteration, message size and more. Read is available only in RC connection mode (as specified in IB spec). For further information, please refer to the tool's man page.
ib_read_lat	Calculates the latency of RDMA read operation of message_size between a pair of machines. One acts as a server and the other as a client. They perform a ping pong benchmark on which one side RDMA reads the memory of the other side only after the other side have read his memory. Each of the sides samples the CPU clock each time they read the other side memory , in order to calculate latency. Read is available only in RC connection mode (as specified in IB spec). For further information, please refer to the tool's man page.
ib_send_bw	Calculates the BW of SEND between a pair of machines. One acts as a server and the other as a client. The server receive packets from the client and they both calculate the throughput of the operation. The test supports features such as Bidirectional, on which they both send and receive at the same time, change of MTU size, tx size, number of iteration, message size and more. Using the "-a" provides results for all message sizes. For further information, please refer to the tool's man page.

Utility	Description
ib_send_lat	Calculates the latency of sending a packet in message_size between a pair of machines. One acts as a server and the other as a client. They perform a ping pong benchmark on which you send packet only if you receive one. Each of the sides samples the CPU each time they receive a packet in order to calculate the latency. Using the "-a" provides results for all message sizes. For further information, please refer to the tool's man page.
ib_write_bw	Calculates the BW of RDMA write between a pair of machines. One acts as a server and the other as a client. The client RDMA writes to the server memory and calculates the BW by sampling the CPU each time it receives a successful completion. The test supports features such as Bidirectional, in which they both RDMA write to each other at the same time, change of MTU size, tx size, number of iteration, message size and more. Using the "-a" flag provides results for all message sizes. For further information, please refer to the tool's man page.
ib_write_lat	Calculates the latency of RDMA write operation of message_size between a pair of machines. One acts as a server and the other as a client. They perform a ping pong benchmark on which one side RDMA writes to the other side memory only after the other side wrote on his memory. Each of the sides samples the CPU clock each time they write to the other side memory, in order to calculate latency. For further information, please refer to the tool's man page.
raw_ethernet_bw	Calculates the BW of SEND between a pair of machines. One acts as a server and the other as a client. The server receive packets from the client and they both calculate the throughput of the operation. The test supports features such as Bidirectional, on which they both send and receive at the same time, change of MTU size, tx size, number of iteration, message size and more. Using the "-a" provides results for all message sizes. For further information, please refer to the tool's man page.
raw_ethernet_lat	Calculates the latency of sending a packet in message_size between a pair of machines. One acts as a server and the other as a client. They perform a ping pong benchmark on which you send packet only if you receive one. Each of the sides samples the CPU each time they receive a packet in order to calculate the latency. Using the "-a" provides results for all message sizes. For further information, please refer to the tool's man page.

3.6 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 NVIDIA representative or NVIDIA Support at networking-support@nvidia.com.

.

The chapter contains the following sections:

3.6.1 General 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.
NVIDIA 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.
NVIDIA adapters are not installed in the system.	Misidentification of the NVIDIA adapter installed	Run the command below and check NVIDIA's MAC to identify the NVIDIA adapter installed. `lspci | grep Mellanox' or 'lspci -d 15b3:` **Note**: NVIDIA MACs start with: 00:02:C9:xx:xx:xx, 00:25:8B:xx:xx:xx or F4:52:14:xx:xx:xx"
The default device may vary when invoking user apps (such as ibv_asyncwatch) which run using a specific device.	The default device for such apps is the first device in the device list generated by libibverbs. This first device in the list varies, depending on which and how many InfiniBand devices are installed on the host, which slot the devices are installed on, whether they use SR-IOV, and other factors.	Always specify the desired device explicitly when running userspace apps, by using the provided command line parameter (for example: ibv_asyncwatch -d <dev>).
Insufficient memory to be used by udev upon OS boot.	udev is designed to fork() new process for each event it receives so it could handle many events in parallel, and each udev instance consumes some RAM memory.	Limit the udev instances running simultaneously per boot by adding udev.children-max=<number> to the kernel command line in grub.
Operating system running from root file system located on a remote storage (over NVIDIA devices), hang during reboot/shutdown (errors such as "No such file or directory" will appear).	The openibd service script is called using the 'stop' option by the operating system. This option unloads the driver stack. Therefore, the OS root file system disappears before the reboot/ shutdown procedure is completed, leaving the OS in a hang state.	Disable the openibd 'stop' option by setting 'ALLOW_STOP=no' in /etc/ infiniband/openib.conf configuration file.
NVIDIA adapter warning print to dmesg: `Detected insufficient power on the PCIe slot (xxxW).`	Insufficient PCI power.	Investigate the cause for lack of PCI power.

3.6.2 Ethernet Related Issues

Issue	Cause	Solution
Ethernet interfaces renaming fails leaving them with names such as renameXY.	Invalid udev rules.	Review the udev rules inside the "/etc/ udev/rules.d/70-persistent-net.rules" file. Modify the rules such that every rule is unique to the target interface, by adding correct unique attribute values to each interface, such as dev_id, dev_port and KERNELS or address). Example of valid udev rules: `SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", ATTR{dev_port}=="0", KER- NELS=="0000:08:00.0", NAME="eth4" SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", ATTR{dev_port}=="1", KER- NELS=="0000:08:00.0", NAME="eth5"`
No link.	Misconfiguration 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 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`

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 breakout cable side is connected to the SwitchX.

3.6.3 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.
Unable to stop the driver with the following on screen message: `ERROR: Module <module> is in use`	An external application is using the reported module.	Manually unloading the module using the '`modprobe -r`' command.

Issue	Cause	Solution
Logical link fails to come up while port logi- cal 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 host connected to the fabric. 2. If SM is not running, activate the SM on a node or on managed switch.
InfiniBand utilities commands fail to find devices on the system. For example, the 'ibv_devinfo' command fail with the following output: Failed to get IB devices list: Function not implemented	The InfiniBand utilities commands are invoked when the driver is not loaded.	Load the driver: `/etc/init.d/openibd start`

3.6.4 Installation Related Issues

3.6.4.1 Installation Issues

Issue	Cause	Solution
Driver installation fails.	The install script may fail for the following reasons: • Using an unsupported installation option • Failed to uninstall the previous installation due to dependencies being used • The operating system is not supported • The kernel is not supported. You can run mlnx_add_kernel_sup- port.sh in order to to generate a MLNX-OFED package with drivers for the kernel • Required packages for installing the driver are missing • Missing kernel backport support for non supported kernel	• Use only supported installation options. The full list of installation options case be displayed on screen by using: `mlnxofedinstall --h` • Run 'rpm -e' to display a list of all RPMs and then manually uninstall them if the preliminary uninstallation failed due to dependencies being used. • Use a supported operating system and kernel • Manually install the missing packages listed on screen by the installation script if the installation failed due to missing prerequisites.
After driver installation, the openibd service fail to start. This message is logged by the driver: Unknown symbol	The driver was installed on top of an existing In-box driver.	1. Uninstall the MLNX_OFED driver. `ofed_uninstall.sh` 2. Reboot the server. 3. Search for any remaining installed driver. If found, move them to the /tmp directory from the current directory. 4. Re-install the MLNX_OFED driver. 5. Restart the openibd service.

3.6.4.2 Fixing Application Binary Interface (ABI) Incompatibility with MLNX_OFED Kernel Modules

⚠ This section is relevant for RedHat and SLES distributions only.

3.6.4.2.1 Overview

MLNX_OFED package for RedHat comes with RPMs that support KMP (weak-modules), meaning that when a new errata kernel is installed, compatibility links will be created under the weak-updates directory for the new kernel. Those links allow using the existing MLNX_OFED kernel modules without the need for recompilation. However, at times, the ABI of the new kernel may not be compatible with the MLNX_OFED modules, which will prevent loading them. In this case, the MLNX_OFED modules must be rebuilt against the new kernel.

3.6.4.2.1.1 Detecting ABI Incompatibility with MLNX_OFED Modules

When MLNX_OFED modules are not compatible with a new kernel from a new OS or errata kernel, no links will be created under the weak-updates directory for the new kernel, causing the driver load to fail. Checking for the existence of needed module links under weak-updates directory can be done by reloading the MLNX_OFED modules. If one or more modules are missing, the driver reload will fail with an error message.

Example:

```
********************************************************************************
# /etc/init.d/openibd restart
Unloading HCA driver:                                      [  OK  ]
Loading HCA driver and Access Layer:                       [  OK  ]
Module rdma_cm belong to kernel which is not a part of MLNX[FAILED]kipping...
Loading rdma_ucm                                           [FAILED]
********************************************************************************
```

3.6.4.2.1.2 Resolving ABI Incompatibility with MLNX_OFED Modules

In order to fix ABI incompatibility with MLNX_OFED modules, the modules should be recompiled against the new kernel, using the mlnx_add_kernel_support.sh script, available in MLNX_OFED installation image.
There are two ways to recompile the MLNX_OFED modules:

1. Local recompilation and installation on one server.
   Run the `mlnxofedinstall` command to recompile the kernel modules and reinstall the whole MLNX_OFED on the server. Mount MLNX_OFED ISO image or extract the TGZ file:

   ```
   # cd <MLNX_OFED dir>
   # ./mlnxofedinstall --skip-distro-check --add-kernel-support --kmp --force
   ```

   Notes:
   - The `--kmp` flag will enable rebuilding RPMs with KMP (weak-updates) support for the new kernel. Therefore, in the next OS/kernel update, the same modules can be used with the new kernel (assuming that the ABI compatibility was not broken again).

- The command above will rebuild only the kernel RPMs (using mlnx_add_kernel_support.sh), and will save the resulting MLNX_OFED package under /tmp and start installing it automatically. This package can be used for installation on other servers using regular `mlnxofedinstall` command or yum.

2. Preparing a new image on one server and deploying it on the cluster.
    a. Use the `mlnx_add_kernel_support.sh` script directly only to rebuild the kernel RPMs (without running any installations) on one server. Mount MLNX_OFED ISO image or extract the TGZ file:

```
# cd <MLNX_OFED dir>
# ./mlnx_add_kernel_support.sh -m $PWD --kmp -y
```

Note: This command will save the resulting MLNX_OFED package under /tmp.
Example:

```
*******************************************************************************
# cd /tmp/MLNX_OFED_LINUX-3.3-1.0.0.0-DB-rhel7.0-x86_64
# ./mlnx_add_kernel_support.sh -m $PWD --kmp -y
Note: This program will create MLNX_OFED_LINUX TGZ for rhel7.1 under /tmp directory.
See log file /tmp/mlnx_ofed_iso.23852.log

Building OFED RPMS . Please wait...
Creating metadata-rpms for 3.10.0-229.14.1.el7.x86_64 ...
WARNING: Please note that this MLNX_OFED repository contains an unsigned rpms,
WARNING: therefore, you should set 'gpgcheck=0' in the repo conf file.
Created /tmp/MLNX_OFED_LINUX-3.3-1.0.0.0-rhel7.1-x86_64-ext.tgz
*******************************************************************************
```

    b. Install the newly created MLNX_OFED package on the cluster:
    Option 1: Copy the package to the servers and install it using the `mlnxofedinstall` script.
    Option 2: Deploy the MLNX_OFED package using YUM (for YUM installation instructions, refer to Installing MLNX_OFED Using YUM section):
        i. Extract the resulting MLNX_OFED image and copy it to a shared NFS location.
        ii. Create a YUM repository configuration.
        iii. Install the new MLNX_OFED kernel RPMs on the servers: `# yum update`
        Example:

```
*******************************************************************************
...
...
=============================================================================================================
=====================
Package                 Arch     Version
Repository  Size
=============================================================================================================
=====================
Updating:
epel-release            noarch
7-7                                                                 epel        14 k
kmod-iser               x86_64   1.8.0-OFED.3.3.1.0.0.1.gf583963.201606210906.rhel7u1
mlnx_ofed   35 k
kmod-isert              x86_64   1.0-OFED.3.3.1.0.0.1.gf583963.201606210906.rhel7u1
mlnx_ofed   32 k
kmod-kernel-mft-mlnx    x86_64   4.4.0-1.201606210906.rhel7u1
mlnx_ofed   10 k
kmod-knem-mlnx          x86_64   1.1.2.90mlnx1-OFED.3.3.0.0.1.0.3.1.ga04469b.201606210906.rhel7u1
mlnx_ofed   22 k
kmod-mlnx-ofa_kernel    x86_64   3.3-OFED.3.3.1.0.0.1.gf583963.201606210906.rhel7u1
mlnx_ofed  1.4 M
kmod-srp                x86_64   1.6.0-OFED.3.3.1.0.0.1.gf583963.201606210906.rhel7u1
mlnx_ofed   39 k

Transaction Summary
=============================================================================================================
=====================
Upgrade  7 Packages
...
...
*******************************************************************************
```

Note: The MLNX_OFED user-space packages will not change; only the kernel RPMs will be updated. However, “YUM update” can also update other inbox packages (not related to OFED). In order to install the MLNX_OFED kernel RPMs only, make sure to run:

```
# yum install mlnx-ofed-kernel-only
```

Note: mlnx-ofed-kernel-only is a metadata RPM that requires the MLNX_OFED kernel RPMs only.

c. Verify that the driver can be reloaded:

```
# /etc/init.d/openibd restart
```

3.6.5 Performance Related Issues

Issue	Cause	Solution
The driver works but the transmit and/or receive data rates are not optimal.	-	These recommendations may assist with gaining immediate improvement: 1. Confirm PCI link negotiated uses its maximum capability 2. Stop the IRQ Balancer service: `/etc/init.d/irq_balancer stop` 3. Start mlnx_affinity service: `mlnx_affinity start` For best performance practices, please refer to the "Performance Tuning Guide for NVIDIA Network Adapters".
Out of the box throughput performance in Ubuntu14.04 is not optimal and may achieve results below the line rate in 40GE link speed.	IRQ affinity is not set properly by the irq_balancer	For additional performance tuning, please refer to Performance Tuning Guide.

3.6.6 SR-IOV Related Issues

Issue	Cause	Solution
When assigning a VF to a VM the following message is reported on the screen: `PCI-assgine: error: requires KVM support`	SR-IOV and virtualization are not enabled in the BIOS.	1. Verify they are both enabled in the BIOS 2. Add to the GRUB configuration file to the following kernel parameter: "intel_immun=on" (see “Setting Up SR-IOV” section).

3.6.7 PXE (FlexBoot) Related Issues

Issue	Cause	Solution
PXE boot timeout.	The '`always-broadcast`' option is disabled.	Enable '`always-broadcast on`'. For the complete procedure, please refer to Linux PXE User Guide.
PXE InfiniBand link fails with the following messages although the DHCP request was sent: *Initializing* and *The socket is not connected*.	Either the SM is not running in the fabric or the SM default multicast group was created with non-default settings.	1. Activate the SM on a node or on managed switch. 2. Check in the SM partitions.conf file that the default partition rate and MTU setting are SDR and 2K, respectively. The PXE is establishing by default an SDR link set with an MTU of 2K. If the default multicast group opened with different rate and/or MTU, the SM will deny the PXE request to join.
NVIDIA adapter is not identified as a boot device.	The expansion ROM image is not installed on the adapter. or the server's BIOS is not configured to work on Legacy mode	1. Run a flint query to display the expansion ROM information. For example: "flint -d /dev/mst/mt4099_pci_cr0 q" and look for the "Rom info:" line. For further information on how to burn the ROM, please refer to MFT User Manual. 2. Make sure the BIOS is configured to work in Legacy mode if the adapter's firmware does not include a UEFI image.

3.6.8 RDMA Related Issues

Issue	Cause	Solution
Infiniband-diags tests, such as 'ib_write_bw', fail between systems with different driver releases.	When running a test between 2 systems in the fabric with different Infiniband-diags packages installed.	Run the test using the same perftest RPM on both systems.

3.6.9 Debugging Related Issues

Issue	Cause	Solution
False positive errors when running applications with valgrind.	Default MLNX_OFED libraries are compiled with- out valgrind support and several resources are managed by the kernel.	Libraries' files compiled with valgrind support are installed under " `/usr/ lib64/mlnx_ofed/valgrind/` " • To run an application over these libraries, thus prevent false positive errors: `# env LD_LIBRARY_PATH=/usr/ lib64/ mlnx_ofed/valgrind/ val- grind [valgrind options] <application cmd>` • To suppress most of valgrind's false positive errors, generate the suppression file: `#./generate_mlnx_ofed_- supp.sh > mlnx.supp`

3.6.10 OVS Offload Using ASAP2 Direct Related Issues

Issue	Cause	Solution(s)
Traffic is not offloaded	OVS uses TC flower classifier to add offloading rules to both the software and the hardware. • TC flower classifier fails to add a rule. • A rule was added to the TC flower classifier but failed to be added to the firmware.	• Check for system error in dmesg or the system logging facility like journalctl • Check OVS logs for errors • Dump the rules using the TC command line For example: Dump rules on a specific interface `# tc filter show dev ens4f0 parent ffff:`

4 Common Abbreviations and Related Documents

Common Abbreviations and Acronyms

Abbreviation/ Acronym	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
RoCE	RDMA over Converged Ethernet
SL	Service Level
SRP	SCSI RDMA Protocol
MPI	Message Passing Interface
QoS	Quality of Service
ULP	Upper Layer 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*.

Term	Description
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	An 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
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
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 the agency of the master SM
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 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)	An NVIDIA technology that allows NVIDIA channel adapter devices (ConnectX®) to simultaneously connect to an InfiniBand subnet and a 10GigE subnet (each subnet connects to one of the adapter ports)

Related Documentation

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 NVIDIA adapter devices	See the Release Notes relevant to your adapter device
MFT User Manual and Release Notes	NVIDIA Firmware Tools (MFT) User Manual and Release Notes documents

Document Name	Description
WinOF User Manual	Mellanox WinOF User Manual describes the installation, configuration, and operation of NVIDIA Windows driver
VMA User Manual	NVIDIA VMA User Manual describes the installation, configuration, and operation of NVIDIA VMA driver

5 Documentation History

5.1 Release Notes History

5.1.1 Changes and New Features History

This section includes history of changes and new feature of three major (GA) releases back. For older versions' history, please refer to their dedicated release notes.

Supported Cards	Description
All HCAs	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-4 / ConnectX -4 Lx / ConnectX-5 / ConnectX-6 / ConnectX-6 Dx / ConnectX-6 Lx / ConnectX-7 / BlueField-2
ConnectX-6 Dx and above	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-6 Dx / ConnectX-6 Lx / ConnectX-7 / BlueField-2
ConnectX-6 and above	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-6 / ConnectX-6 Dx / ConnectX-6 Lx / ConnectX-7 / BlueField-2
ConnectX-5 and above	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-5 / ConnectX-6 / ConnectX-6 Dx / ConnectX-6 Lx / ConnectX-7 / BlueField-2
ConnectX-4 and above	Supported in the following adapter cards **unless specifically stated otherwise:** ConnectX-4 / ConnectX -4 Lx / ConnectX-5 / ConnectX-6 / ConnectX-6 Dx / ConnectX-6 Lx / ConnectX-7 / BlueField-2

Feature/Change	Description
23.10-5.1.4.0	
General	
General	Bug fixes

Feature/Change	Description
23.10-4.0.9.1	
General	
Embedded Components	Updated the versions of the following embedded component: • MFT • ConnectX adapter cards firmware For further information, see Embedded Components section.
General	Bug fixes

Feature/Change	Description
23.10-3.2.2.0	
Operating Systems	Added support for the following OSes: • RHEL 8.10 • RHEL 9.4 • SLES15-SP6 • Debian 12.5 For further information, see Supported Operating Systems section
Embedded Components	Updated the versions of the following embedded component: • MFT • ConnectX adapter cards firmware For further information, see Embedded Components section.
Bridge Offload	Enabled Bridge Offload in mlx5_core by default. A new option was added to the configure of mlnx_ofa_kernel to disable it: `"--without-bridge-offload"`. To use this option from `"mlnxofedinstall --add-kernel-support"` (kernel modules rebuilding), add the following to the command line: `"--kernel-extra-args --without-bridge-offload"`.
General	Bug fixes

Feature/Change	Description
23.10-2.1.3.1	
General	
General	Bug fixes

Feature/Change	Description
23.10-0.5.5.0	
ASAP^2 Features	
Rate Limiter Extension for LAG Mode	[ConnectX-5 and above] During rate configuration, this feature considers the aggregated port speed when LAG is configured, so the functions (PF,VF,SF) can utilize the maximum aggregated link speed for transmission.
Core Features	
Live Migration Support with IPsec Full Offload	[ConnectX-7 and above] Added an option to perform live migration on a machine with IPsec enabled. This allows having a VF on VM that has a valid IPsec state and migrate it without having to remove the IPsec. **Note:** This feature is currently not supported when working in MPV mode (Multi port VHCA/Dual port). Such support will be added in future releases.
Exposing the Max SF Configuration via Devlink Resource	[All HCAs] This feature allows user applications to view the maximum number of SFs (Scalable Functions) configured on the system via general Linux API, Devlink resource.
NetDev Features	

NIC Temperature Exposure	[ConnectX-5 and above] Added support for NIC temperature exposure by implementing the hwmon kernel API.
Counters Indicating Packet Drops due to Catastrophic Steering Errors	[All HCAs] Added two counters through the Devlink health reporter - generated_pkt_steering_fail and handled_pkt_steering_fail. These counters indicate packet drops due to catastrophic steering errors.
RDMA Features	
Using the RDMA Tool to Allow Controlled QKEYs Use by Non-privileged Users	[All HCAs] To allow using controlled QKEYs by non-privileged users, the following command can now be used in the iproute2 RDMA tool, instead of the "enforce_qkey_check" module parameter, which will be deprecated: `----- $rdma system set privileged-qkey on`
IPsec Configuration Support When All IPsec Operations are Offloaded to the Hardware	[ConnectX-7 and above] Added support for configuring IPsec even when all IPsec operations are completely offloaded to the hardware. This feature not only provides a significant performance improvement, but also enables the use of IPsec over RoCE packets, which are outside the network stack and cannot be used without full hardware offload. As a result, users can now leverage the benefits of IPsec protocol with RoCE V2, even when using SR-IOV VFs.
Security Features	
Configuration from VM/VF for Full Offload Transport Mode	[ConnectX-6 Lx and above] This feature introduces two new boolean attributes of a port function: ipsec_crypto and ipsec_packet (also known as IPsec Full Offload). The goal is to provide a level of granularity for controlling VF IPsec offload capabilities similar to what is currently offered in the software mode. This allows users to decide if they want both types of IPsec offload enabled for a VF, just one of them, or none (which is the default).
General	
General	Bug fixes

Feature/ Change	Description
23.07-0.5.1.2	
RDMA Features	
QKEY Mitigation in the Kernel	QKEY creation with the MSB set is available now for non-privileged users as well. To allow non-privileged users to create QKEY with MSB set, the below new module parameter was added to ib_uverbs module: • **Module Parameter:** `enforce_qkey_check` • **Description:** Force QKEY MSB check for non-privileged user on UD QP creation • **Default:** 0 (disabled) **Note:** In this release, this module parameter is disabled by default to ensure backward compatibility and give customers the opportunity to update their applications accordingly. In the upcoming release, it will be enabled by default, and later on deprecated.
23.07-0.5.0.0	
ASAP^2 Features	
Header Rewrite and CT Support	[BlueField-2] Added support for offloading CT rules with header rewrite of L3.

CT UDP Unidirectional Traffic Offload	[All HCAs] Added support for offloading long-running unidirectional UDP connection with Conntrack.
CT Rules with NAT and Mirror Offload	[BlueField-2] Added support for offloading CT rules with NAT and mirror.
Core Features	
Embedded CPU Virtual Functions	[Bluefield-2] Added the ability to create virtual functions within the ARM.
Full Chip Reset on BlueField-2	[BlueField-2] Full chip reset in DPU mode is now supported on BlueField-2 when using mlxfwreset with the "--sync 1" option. During this flow, the ARM is rebooted and the firmware is reloaded.
Firmware Page Limit via VHCA_ICM_CTRL	[ConnectX-5 and above] Added the ability to expose page limit via the VHCA_ICM_CTRL FW register. The expected behavior is similar to the previous page limit, except for the firmware size limitation. The new max limit is 2^32-1, which represents 'no limit'.
PRE_COPY State Support in the mlx5-vfio Driver	[ConnectX-6 Dx and above] Added support for PRE_COPY migration. The optional PRE_COPY state opens the saving data transfer FD before reaching STOP_COPY, and thereby helps reducing the downtime of the VM.
Lightweight Local SFs	[ConnectX-5 and above] Added a feature that decreases the amount of time needed to probe and configure SFs (sub-functions) by reducing the time of Devlink reload of the SF.
QEMU Live Migration Support	[ConnectX-6 Dx and above] Added support for QEMU VM migration with an assigned VF from one source host to another destination host. This is achieved as part of the general QEMU migration flow, which involves suspending the VF on the source host, transferring all its data to the destination host, and resuming the VF on the destination. **Note:** This migration feature includes basic functionality only, and does not yet support advanced features such as dirty page tracking or pre-copy.
4 Ports VF LAG Support	[ConnectX-7] Added support for VF LAG over 4 ports HCAs. **Note:** Only LAGs with all HCA ports are supported. Meaning, with a 4-port HCA, only 4 port LAG is supported. A 2-port or a 3-port LAG is not supported.
Installation	
FIPS Compliance	[All HCAs] Starting from v23.07, OFED is FIPS (Federal Information Processing Standard) compliant for the RH8.8, RH9.2 and SLES15.5 operating systems. The following limitations should be considered: • OFED Installation with FIPS support requires a manual firmware upgrade. • OFED Installation with add-kernel-support will result in the removal of the FIPS support.
NetDev Features	
Enhanced CQE Compression	[ConnectX-5 and above] Added support for the enhanced version of the RX CQE compression hardware feature. By compressing the RX CQEs, the PCI bandwidth utilization is improved and the load on it is reduced. The enhanced version of this device feature has improved latency and CPU utilization.
256 Bit Keys with kTLS Offload Support	[BlueField-2] Added support for kTLS offload with a 256 bit key size.
Improving Affinity Hints according to Numa Distances	[All HCAs] Updated the binary NUMA preference for the system to consider actual distances. Following the update, remote NUMA nodes with shorter distances are preferred over further ones, rather than relying solely on local/remote distinctions.

Flow Steering Decoupling	[All HCAs] Added the ability to decouple flow steering into a separate module, making it loosely coupled and thus easier to read, maintain and debug.
RDMA Features	
IPv6 Address Support	[All HCAs] Reduced the storage IO latency that occurred when establishing a large number of RDMACM (RDMA Connection Manager) connections by setting the RDMACM RoCE static rate to 0.
Reducing Storage IO Latency for a Large Number of RDMACM Connections	[All HCAs] Reduced the storage IO latency that occurred while establishing a large number of RDMACM (RDMA Connection Manager) connections by setting the RDMACM RoCE static rate to 0.
QKEY Mitigation in the Kernel	[All HCAs] Non-privileged users are now blocked from setting controlled/privileged QKEYs (QKEY with MSB set). To allow non-privileged users to create a QKEY with an MSB set, a new module parameter was added to the ib_uverbs module: • module parameter: enforce_qkey_check • Description: Force a QKEY MSB check for non-privileged users on UD QP creation • Default is 0 (disabled) In this release, this module parameter is introduced in disabled state in order to ensure backward compatibility and allow the required applications updates. In the upcoming release, this feature will be enabled by default, and later on deprecated.
Security	
Fast Update Encryption Key Support	[ConnectX-6 Dx and above] Enhanced the performance of DEK operations by introducing a DEK pool that serves the same key purpose and utilizes bulk allocation, destruction and invalidation provided by the firmware. Users can now retrieve a DEK object from the pool, and update it with a key using the modify_DEK command.
Software Steering	
Matching over BTH Acknowledge Bit Support	[Connect-6 DX, Connect-7 and BlueField-2] Added support for matching over BTH acknowledge bit using the mlx5dv_dr API.
General	Bug fixes

Feature/ Change	Description
23.04-1.1.3.0	
Operating Systems	Added support for the following operating systems: • RHEL 9.2 • RHEL 8.8 • SLES15-SP5
23.04-0.5.3.3	
ASAP^2 Features	

VXLAN GBP Options Offload with OVS	[ConnectX-6 Dx and above] Added support of encap and decap of VXLAN tunnel with GBP options offload with OVS.
TC Rules: Additional Actions	[ConnectX-5 and above] Added support for adding TC rules with trap action with additional actions (mirror and pedit).
Multiport E-Switch	[ConnectX-6 Dx] Added support for Multiport E-Switch, a mode where a single E-Switch connects all VPorts and physical ports on the NIC. This allows for scenarios such as sending traffic from a VF created on PF0 to an uplink that is natively associated with the uplink of PF1.
OVS Offload with MACVLAN Interface Above Bond	[ConnectX-6 Dx and above] Added support for OVS offload when MACVLAN interface above bond is attached to OVS bridge.
Core Features	
PCC fwtrace	[ConnectX-6 Dx and BlueField-2] Added support for installing a special user image into the firmware, which can be burned into either of two available slots for such applications, which enables monitoring the image's activities using the fw_tracer located inside the mlx5 driver. To view the output of the tracer, the user can access trace point, but it is important to note that they can only view traces that are generated after enabling the trace point by using the following command: 'echo 1 > sys/kernel/debug/tracing/events/mlx5/mlx5_fw/enable'." **Note:** Not supported in ConnectX-6 Lx adapter cards.
Relaxed Ordering in VFs/QEMU Assigned VFs	[ConnectX-6 and above] Added support for using Relaxed Ordering in VFs directly and in VFs assigned to QEMU. Relaxed Ordering can significantly improve performance on certain setups and, until now, it could be used only in PFs.
Migratable Bit	[ConnectX-6 Dx and above] Added support for Migratable Bit in Live Migration. Because some features cannot be migrated, such as IPsec, for example, when VF is marked as migratable, those features are disabled. This feature allows the user to configure whether VF can be migrated.
Live Migration Dirty Page Tracking Support in Linux	[ConnectX-7] The dirty pages tracking support enables reducing downtime upon live migration. Once it is used, only the pages that were really dirtied by the device will be marked in QEMU as dirty and will be sent to the target upon stop. Without dirty tracking, all RAM is marked dirty so all RAM is resent upon stop and the downtime is increased.
NetDev Features	
Configuring Hairpin Queue Size	[ConnectX-5 and above] Added the ability to configure the number and size of hairpin queues through devlink param command. For example: devlink dev param set pci/0000:08:00.0 name hairpin_queue_size value 512 cmode driverinit devlink dev param set pci/0000:08:00.0 name hairpin_num_queues value 1 cmode driverinit
RDMA Features	
CC - RTT response SL (CNP SL)	[ConnectX-6 Dx and above] Users can now customize the DSCP value of RTT Response packets (in Ethernet, using debugfs). This feature allows for prioritization of RTT Response packets, preventing any delay that might lead to incorrect congestion assumptions on the RTT Requester side.

Expose VF RoCE Statistics on Host Side (Representor)	[All HCAs] Added support for allowing the host to track various statistics for the VFs, specifically all of the Q_counters stats even if he gives the VF to a VM, through the representor Q_counters for the VF which are now exposed over the host.
Fatal QP Error Logging	[All HCAs] With this feature, a kernel error log is now generated when certain fatal QP errors occur.
Enabling Selective Repeat by Default	[ConnectX-7 and above] Added support for making Selective Repeat (SR) protocol to be anabled by default. SR retransmits only the frame that is damaged and not all of the frames that were sent. As such, SR makes more efficient use of network bandwidth.
Software Steering	
Matching IB BTH in RoCE Packets	[ConnectX-6 Dx and above] Added support for matching the IB BTH in RoCE packets. One example of this match is that the user can monitor RoCEv2 CNP (Congestion Notification Packet) by matching BTH opcode 0x81.
General	Bug fixes

Feature/ Change	Description
5.9-0.5.6.0	
ASAP^2 Features	
Linux Bridge VLAN Filtering of 802.1 Q Packets	[ConnectX-6 Dx] Extended mlx5 Linux bridge VLAN offload to support packets tagged with 802.1 Q VLAN ethertype.
Offloading sFlow Sampling Rules	[ConnectX-5 and above] Added support for sFlow sampling rules offloads. sFlow is an industry standard technology for monitoring high speed switched networks. Open vSwitch integrated sFlow to extend the visibility into virtual servers, ensuring data center visibility and control.
Core Features	
Configuring Shared Buffer Size	[ConnectX-6 Dx and above] Enabled user to control shared buffer size and configuration, implicitly. As with each port buffer command the user triggers, the shared buffer configuration will be updated accordingly by the driver.
Control SF Class	[All HCAs] Added support for Control SF Class. Each PCI, PF, VF, SF function, by default, has netdevice, RDMA, and vdpa-net devices always enabled. This feature enables the user to control which device functionality to enable/disable. **Note:** Requires kernel 5.18 or higher.
Installation Features	
ip2gid Tool	[All HCAs] Added support for ip2gid tool. This tool does the following: 1. Resolves a destination IP into a destination GID needed when running a rdmacm applications (ip2gid). 2. Resolves a GID into one PR (PathRecord) or multiple PRs if needed (gid2lid). This tool is needed when rdmacm is used to initiate InfiniBand traffic between nodes on different IP subnets in InfiniBand fabrics.

NetDev Features	
Support RSS over XSK Queues	[All HCAs] Use default RSS functionality to spread traffic across different XSK queues instead of having to provide explicit steering rules.
TLS TIS Pool	[TLS-Enabled Devices] Per-connection hardware TIS objects is used to maintain the device TLS TX context. Use a SW TIS pool for recycling the TIS objects instead of destroying/ creating them. This reduces the interaction with the device via the FW command interface, which increases the TLS connection rate.
RDMA Features	
Expand Rep Counters	[ConnectX-5 and above] Adding RDMA traffic-only counters for rep devices. These counters can now be read from host with ethtool or from sysfs and not only from the cointainer.
UMR QP Recilency	[ConnectX-5 and above] Added a recovery flow for the driver's UMR logic so that other UMR requests can be proccessed after the error UMR was dropped and the UMR QP was reset. Previously, a faulty UMR request would have moved the QP to error state and disable any option to continue issuing UMRs.
General	Bug fixes

Feature/ Change	Description
5.8-1.1.2.1	
General	Bug fixes
5.8-1.0.1.1	
Remove Dependency Between SR-IOV and eSwitch Mode	[All HCAs] Removed dependency between SR-IOV and eSwitch mode. Currently, there are three eSwitch modes: none, legacy, and switchdev (non of which are the default mode). When disabling SR-IOV, the current eSwitch mode will be changed to none. This feature removes eSwitch mode none and also removes dependency between SR-IOV and eSwitch mode.
DevLink Parallel Command	[All HCAs] Added support for running DevLink commands in parallel on different DevLink devices is possible. For example, burning firmware on a few cards on the same host in parallel using DevLink API is now possible.
Graceful Shutdown of Parent and Page Supplier	[All HCAs] Set default graceful period values for functions based on their type. ECPFs will get graceful period of 3 minutes, PFs get 1 minute, and VFs/SFs get 30 seconds.
N Pulses Per Second (NPPS)	[ConnectX-6 Dx and above] Enhanced NPPS to allow setting a pulse period higher than 1 pulse per second and to allow setting the pulse width. If the width is unset, the driver implicitly sets it to half the given period (the width should be less than the pulse period). In this release, the pulse duration ranges between 65536 NS- 524288 NS.
Remote Invalidate Option for MKeys	[All HCAs] Added support for the option to enable remote invalidation when creating a new mkey. This way the rkey for a memory region can be changed frequently.

GPUDirect Over DMA-BUF	[All HCAs] Added support for GPUDirect support over dma-buf. As such, using the new mechanism nv_peer_mem is no longer required. The following is required for dma-buf support: • Linux kernel version 5.12 or later • OpenRM version 515 or later Perftest support was added as well: Default option in perftest is without dmabuf. To run with this option, add --use_cuda_dmabuf in addition to use_cuda flag.
Floating LID	[ConnectX-7] Added support for Floating LID (FLID) which can be used to identify a group of InfiniBand routers that allow communication with another subnet's entity. With this feature, multiple routers can be used per destination so that adaptive routing is supported. The FLID feature needs support from components such as the host, the subnet manager, the router, and more. This feature is only supported on the host portion of the sysrem.
General	Bug fixes

Feature/ Change	Description
5.7-1.0.2.0	
Support Representor Metering Over SFs	[ConnectX-6 Dx and above and BlueField-2] Extended the support of representor metering from supporting only VFs representor to also supporting SFs representor.
Exposing Error Counters on a VPort Manager	[ConnectX-4 and above] Added support for exposing error counters on a VPort manager function for all other VPorts. These counters can be used to detect malicious users who are exploiting flows that can slow the device. The counters are exposed through debugfs under: /sys/kernel/debug/mlx5/esw/<func>/ vnic_diag/
Ethtool RX Flow Steering for IPoIB	[All HCAs] Enabled steering of IPoIB packets via Ethtool, in the same way it is done today for Ethernet packets.
Memory Consumption Minimization	[ConnectX-4 and above] Added support for providing knobs which enable users to minimize memory consumption of mlx5 functions (PF/VF/SF).
XDP Support for Uplink Representors	[ConnectX-5 and ConnectX-6 Dx) Added XDP support for uplink representors in switchdev mode.
Resiliency to tx_port_ts	[ConnectX-6 Dx and above] Added resiliency to the tx_port_ts feature. private-flag may be enabled via ethtool tx_port_ts which provides a more accurate time-stamp. In very rare cases, the said time-stamp was lost, leading to losing the synchronization altogether. This feature allows for fast recovery and allows to quickly regain synchronization.
Database of Devlink Health Asserts	[ConnectX-4 and above] Health buffer now contains more debug information like the epoch time in sec of the error and the error's severity. The print to dmesg is done with the debug level corresponding to the error's severity. This allows the user to use dmesg attribute: dmesg --level to focus on different severity levels of firmware errors.

Feature/ Change	Description
5.7-1.0.2.0	
Expose FEC Counters via Ethtool	[ConnectX-5 and above] Exposed the following FEC (forward error detection) counters: ETHTOOL_A_FEC_STAT_CORRECTED • fc_fec_corrected_blocks_laneX • rs_fec_corrected_blocks ETHTOOL_A_FEC_STAT_UNCORR • fc_fec_uncorrectable_blocks_laneX • rs_fec_uncorrectable_blocks ETHTOOL_A_FEC_STAT_CORR_BITS • phy_corrected_bits Command: ethtool -I show-fec <ifc>
Application Device Queues	[ConnectX-4 and above] Added driver-level support for Application Device Queues. This feature allows partition defining over the RX/TX queues into groups and isolates traffic of different applications. This mainly improves predictability and tail latency.
Reinjection of Packets Into Kernel	[All HCAs] Added support for a new software steering action, mlx5dv_dr_action_create_dest_root_table(). This action can be used to forward packets back into a level 0 table. As a table with level 0 is the kernel owned table, this will result in injecting packets to the kernel steering pipeline.
DCT LAG	[ConnectX-6 Dx and above] Added firmware support to allow explicit port selection based on steering and not QP affinity. Functionality: 1. Use LAG Hash Mode for the HCA with two ports, if supported. 2. Keep port affinity function in LAG Hash Mode if it supports bypass select flow table in non-SwitchDev mode.
AES-XTS in RDMA	Added support for plaintext AES-XTS DEKs.
General	Bug fixes

5.1.1.1 Customer Affecting Changes

Feature/Change	Description
23.10-1.1.9.0	
Lightweight Local SFs	Following the addition of the Lightweight Local SFs feature in version 23.07, in order to configure the scalable-functions, follow the revised instructions as detailed in the Step-by-Step Guide. **Note:** "Step 2.9 - Set all SF specific device parameters" is now mandatory for local SFs.

23.10-0.5.5.0	
Customer Affecting Change	**Description**

Debugfs Directory Path Change	The debugfs directory of each interface can now be found under: /sys/kernel/debug/mlx5//, and not directly under the root of the debugfs filesystem (/sys/debug/kernel).
Deprecation of OFED Public Power PC Installation	Starting from this release, MLNX_OFED releases for Power PC are no longer available for download from the public Download Center web page. Instead, you can find it on the following page: https://network.nvidia.com/support/firmware/ibm-systemp/.
Pre-notification: Deprecation of Older Operating Systems	Starting from next release, MLNX_OFED releases will no longer support operating systems with kernels below v4.18. This includes the following systems: • Rhel7.x • Debian9.13 • Sles12.x • Xenserver7.1
Customer Affecting Change	**Description**
23.07-0.5.1.2	
Creating a QKEY with an MSB Set	To allow non-privileged users to create a QKEY with an MSB set, a new module parameter was added. For details, please see "QKEY Mitigation in the Kernel" under New Features.
23.07-0.5.0.0	
IRQ Naming	IRQ renaming is no longer done when bringing the interface up/down. The IRQ name is now constant and is not affected by the interface state.
RPM Packages Verification Key	RPM_GPG-KEY-Mellanox (or its variants) is no longer the public key that verifies RPM packages of MLNX_OFED. Instead, the RPM_GPG-KEY-Mellanox file on the top-level directory of the ISO should be used.
Hairpin sysfs Support	Hairpin sysfs support was restricted to physical and virtual functions only.
mlx5_core node_guid Module Parameter	Removed a non-functional mlx5_core node_guid module parameter.
OpenSM Init	Starting from this release, the opensm init service moves from init.d (`/etc/init.d/opensmd start`) to systemd (`# service opensmd start`).
Apt Signing Key	Starting from this release, the public key that signed the apt repository of MLNX_OFED is included in the ISO in a format that can be used directly by the apt for repository signatures verification.
IPoIB ULP Mode Deprecation	Starting from this release, MLNX_OFED supports IPoIB enhanced mode only. The ability to switch back to ULP mode using ipoib_enhanced module parameter is not supported. For more information about the enhanced mode, please refer to the OFED user manual, example: Enhanced IP over InfiniBand.
Pre-notification: Deprecation of OFED Public Power PC Installation	Starting from next release, MLNX_OFED releases for Power PC will no longer be available for download from the public Download Center web page.

Customer Affecting Change	Description
23.07-0.5.0.0	
Creating a QKEY with an MSB Set	To allow non-privileged users to create a QKEY with an MSB set, a new module parameter was added. For details, please see “QKEY Mitigation in the Kernel“ under New Features.
IRQ Naming	IRQ renaming is no longer done when bringing the interface up/down. The IRQ name is now constant and is not affected by the interface state.
RPM Packages Verification Key	RPM_GPG-KEY-Mellanox (or its variants) is no longer the public key that verifies RPM packages of MLNX_OFED. Instead, the RPM_GPG-KEY-Mellanox file on the top-level directory of the ISO should be used.
Hairpin sysfs Support	Hairpin sysfs support was restricted to physical and virtual functions only.
mlx5_core node_guid Module Parameter	Removed a non-functional mlx5_core node_guid module parameter.
OpenSM Init	Starting from this release, the opensm init service moves from init.d (`/etc/init.d/opensmd start`) to systemd (`# service opensmd start`).
Apt Signing Key	Starting from this release, the public key that signed the apt repository of MLNX_OFED is included in the ISO in a format that can be used directly by the apt for repository signatures verification.
IPoIB ULP Mode Deprecation	Starting from this release, MLNX_OFED supports IPoIB enhanced mode only. The ability to switch back to ULP mode using ipoib_enhanced module parameter is not supported. For more information about the enhanced mode, please refer to the OFED user manual, example: Enhanced IP over InfiniBand.
Pre-notification: Deprecation of OFED Public Power PC Installation	Starting from next release, MLNX_OFED releases for Power PC will no longer be available for download from the public Download Center web page.

Customer Affecting Change	Description
23.04-0.5.3.3	
Netdev Interface Configuration is not Preserved During Reload/ Reset/ Recovery	As of OFED 23.04, during reset/reload/recovery flows, the netdev interface is destroyed and re-created (rather than just suspended). As a result, the netdev interface configuration is not preserved, and must be re-applied. The way to do this is to use proper network-scripts and/or udev rules files to configure network interface parameters. These are automatically triggered whenever a netdev interface is added, regardless of whether it was added due to a user-initiated operation or an automatic failure recovery operation. Thus, no special processing is required to re-apply the network interface configuration parameters following a reset/reload/recovery operation – it is performed automatically.
Prenotification : Deprecation of OFED Public Power PC Installation	Starting next release, MLNX_OFED releases for Power PC will no longer be available for download from the public Download Center web page.

Customer Affecting Change	Description
23.04-0.5.3.3	
Prenotification : ULP Mode Deprecation	Starting next release, MLNX_OFED will support IPoIB enhanced mode only. The ability to switch back to ULP mode using ipoib_enhanced module param will not be supported. For more information about the enhanced mode, please refer to OFED user manual, example: Enhanced IP over InfiniBand
Installation, ISO, RedHat	In order to address RHEL kernel symbol changes, ISO images for the following operating systems are built with the updated kernel versions as follows: OS Name Old Kernel New Kernel rhel8.6-aarch64 4.18.0-372.9.1.el8_6.aarch64 4.18.0-372.41.1.el8_6.aarch64 rhel8.6-ppc64le 4.18.0-372.0.1.el8_6.ppc64le 4.18.0-372.41.1.el8_6.ppc64le rhel8.6-x86_64 4.18.0-372.9.1.el8_6.x86_64 4.18.0-372.41.1.el8_6.x86_64 rhel8.7-aarch64 4.18.0-425.3.1.el8.aarch64 4.18.0-425.14.1.el8_7.aarch64 rhel8.7-ppc64le 4.18.0-425.3.1.el8.ppc64le 4.18.0-425.14.1.el8_7.ppc64le rhel8.7-x86_64 4.18.0-425.3.1.el8.x86_64 4.18.0-425.14.1.el8_7.x86_64 rhel9.0-aarch64 5.14.0-70.13.1.el9_0.aarch64 5.14.0-70.46.1.el9_0.aarch64 rhel9.0-ppc64le 5.14.0-70.13.1.el9_0.ppc64le 5.14.0-70.46.1.el9_0.ppc64le rhel9.0-x86_64 5.14.0-70.13.1.el9_0.x86_64 5.14.0-70.46.1.el9_0.x86_64 rhel9.1-aarch64 5.14.0-162.6.1.el9_1.aarch64 5.14.0-162.19.1.el9_1.aarch64 rhel9.1-ppc64le 5.14.0-162.6.1.el9_1.ppc64le 5.14.0-162.19.1.el9_1.ppc64le rhel9.1-x86_64 5.14.0-162.6.1.el9_1.x86_64 5.14.0-162.19.1.el9_1.x86_64 This change comes to support RedHat updated kernels without the need to add --add-kernel-support during OFED installation.
Power Setups on UCX/HPC-X	UCX/HPC-X no longer supports Power setups.
NEO-Host	Starting from this release, OFED will discontinue the provision of NEO-Host. NEO-Host can be manually downloaded and installed using the following guide: https://docs.nvidia.com/networking/display/NEOSDKv26/Installation+and+Initial+Configuration#InstallationandInitialConfiguration-DownloadingtheMellanoxNEOSDKSoftware
dapl	Starting from this release, OFED will discontinue the provision of dapl.
Signing Key for SLES15 sp4 and sp5	As of version 23.04, the builds for SLES15 sp4 and sp5 are being singed with a newer signing key. The corresponding public key can be downloaded from https://www.mellanox.com/downloads/ofed/nv_nbu_kernel_signing_key_pub.der instead of https://www.mellanox.com/downloads/ofed/mlnx_signing_key_pub.der.
dump_pr SM Plugin	Starting from this release, OFED will discontinue the provision of dump_pr subnet manager plugin.
mpi-selector	Starting from this release, OFED will discontinue the provision of mpi-selector.
OpenSM Init	Starting 23.07 release, opensm init service will move from init.d to systemd.

Customer Affecting Change	Description
5.9-0.5.6.0	
Deprecation, LAG Mode via Sysfs	Setting LAG mode via Sysfs is going to be deprecated in a future release. Instead, LAG Hash mode will be used by default, similar to upstream behavior.

Customer Affecting Change	Description
5.9-0.5.6.0	
LAG Configuration, PCI Error	From version 5.9, LAG configuration will be lost in case driver incurs a PCI error. Make sure to reconfigure the bond after driver completes the recovery from the PCI error. In releases prior to 5.9, in case of PCI error (EEH injections on PPC setup), the driver recovers LAG bond and reconfigures it automatically in case it what configured before the appearance of the error.

Customer Affecting Change	Description
5.7-1.0.2.0	
Multi-Block Encryption	Multi-block encryption is currently unsupported, due to a hardware limitation.

Feature / Change	Description
5.6-2.0.9.0	
Operating Systems	Added support for the following Operating Systems: RHEL8.6, RHEL9.0, SLES15-SP4.
General	Bug fixes

Feature / Change	Description
5.6-1.0.3.3	
General	
New Adapter Card Support	Added support for ConnectX-7 adapter cards. ConnectX-7 has the same feature set as the ConnectX-6 adapter card.
ASAP^2 Features	
Bridge Spoof Check	[All HCAs] Added support for spoof check with TC flower rules on representors attached to bridge to mirror spoof check SR-IOV functionality.

Setting VF Group Rate Limit	[ConnectX-5 and above] Added support for setting VF group rate limit using Devlink command.
TC Flows on Shared Block	[ConnectX-5 and above] Added support for creation of TC flows on shared block of VF representors.
Flow Metering	[ConnectX-6 Dx and above] Added support for offloading OpenFlow Meters in OVS-DPDK. Please note the following: • Meter offload can be applied only on port 0 and it's VFs • Only one meter per flow is allowed • Only one meter band per meter is allowed • Only meter band type drop is supported • Meter-stats might not be accurate
Core Features	
Firmware Reset	[BlueField-2] Added support of firmware reset in DPU NIC mode.
Increased Robustness of mlx5_core Driver Recovery	[All HCAs] Increased the firmware pre-initialization timeout from 2 minutes to 2 hours when waiting for firmware during driver health recovery, allowing the driver to passively recover from a firmware reset, even if the reset takes an unusually long time. Additionally, added an exit clause to the wait for firmware loop, allowing immediate response to a user initiated device removal.
NetDev Features	
Ethtool CQE Mode Control	[ConnectX-4 and above] Replaced the vendor-specific Ethtool API (priv-flag) with a standard Ethtool API (replaced 'ethtool --set-priv-flags ethX rx_cqe_moder on/off tx_cqe_moder on/off' with 'ethtool -C ethX cqe-mode-rx on/off cqe-mode-tx on/off'). This decreases the amount of vendor-specific configurations and aligns mlx5 driver with the upstream Ethtool API.
SyncE	[ConnectX-6 Dx] Added an indication in SyncE Daemon that states whether SyncE engine moved to holdover state due to failure (the reason for failure will be displayed). In addition, added indication whether SyncE engines collected enough frequency samples in order to move to holdover. **Note:** Not supported in ConnectX-6 Lx adapter cards.
RDMA Features	
VFIO, CQ Interrupt Mode	[ConnectX-5 and above] Added support for VFIO applications to listen on and capture completion events via the Event Queue mechanism.
VFIO, Asynchronous Event	[ConnectX-5 and above] Added support for VFIO applications to listen on and capture device asynchronous events via the Event Queue mechanism.
Security	
OVS-IPSec Full Offload	[BlueField-2] Added support for configuration of IPsec full offload using OVS by adding VXLAN tunnel to OVS with the PSK option.
Software Steering Features	
Full Tunnel Header Matching	[ConnectX-6 Dx and above] Added support for using full-tunnel-header matching along with many other criteria within one matcher. This feature uses the new definer index, defined in the firmware, to build a matcher so that the full tunnel header matching can be used along with all other criteria.

Matching Granularity Change	[ConnectX-5 and above] Added support for matching granularity change. As a result, when creating FDB flow with destination of VPORT, a src_port matching must no longer be added. Now, FDB flow can match all vports and goto a VPORT destination. The new behavior is the same as done on firmware steering.
Installation	
Installation	New options were added to the ofed_uninstall.sh script: - `-only-kernel` and - `-only-user`. Those can be used to uninstall only kernel packages or only user-space packages (the equivalent of kernel-only install or user-space-only install, respectively). This may be useful to keep different sets of kernel and user-space installations. When running the uninstall script with a combination of `--only-kernel` and `--only-user` produced an undefined result.

Customer Affecting Change	Description
Customer Affecting Changes 5.6-1.0.3.3	
Interface Renaming, PF/VF, Udev	The OFED driver no longer performs Ethernet NetDev interface renaming for PFs and VFs. The udev rules file which implemented renaming (82-net-setup-link.rules) and its supporting script vf-net-link-name.sh are no longer installed by default. Renaming is thus performed by underlying mechanisms -- in udev, in the kernel, and in the BIOS. Users who wish to continue using the OFED driver renaming mechanism must add option --copy-ifnames-udev to the OFED install command. To install these files at a later time, copy them from one of the following directories: • /usr/share/doc/mlnx-ofa_kernel (RHEL8 and newer) • /usr/share/doc/mlnx-ofa_kernel-[1-9]* (RHEL 7.X) • /usr/share/doc/packages/mlnx-ofa_kernel (SLES) • /usr/share/doc/mlnx-ofed-kernel-utils/examples (Debian-based releases) ⚠ • File 82-net-setup-link.rules should be copied to directory /etc/udev/rules.d • File vf-net-link-name.sh should be copied to directory /etc/infiniband (make sure that it has both read and execute permission) • After copying over the files, the driver should be restarted for the copied files to take effect • Customers who wish prevent renaming of NetDev names should add "net.ifnames=0 biosdevname=0" to the kernel boot command line, and then reboot the host
Community Operating Systems	Starting OFED 5.6, NVIDIA is introducing a new support model for OFED used on open source community operating systems. The goal of this new support model is to enable customers to use community-maintained variants of the Linux operating system, without being limited to major distributions that NVIDIA provides primary support for. For more information, see "Installation on Community Operating Systems" section in the user manual. For a list of supported Community OSs, please see "Supported Community Operating Systems" section in the release notes.
ar_mgr Subnet Manager Plugin	ar_mgr subnet manager plugin is no longer supported. For adaptive routing and SHIELD subnet manager configuration, please see the MLNX_OFED user manual.
Fabric Collector in UFM	Starting UFM v6.7, Fabric Collector is no longer supported. For more information, see the UFM release notes. ⚠ Please note that UFM is no longer bundled with OFED.

Customer Affecting Change	Description
Customer Affecting Changes 5.6-1.0.3.3	
OVS-DPDK—Partial Offload	Starting OFED 5.6, OVS-DPDK does not support partial offload.

5.1.2 Bug Fixes History

This table lists the bugs fixed in the last three major GA releases. For a list of old bug fixes, please refer to the release notes of the desired version.

Below are the bugs fixed in this version. For a list of fixes previous version, see Bug Fixes History.

Internal Reference Number	Description
4140272 / 4496401 / 4567694 / 4567695	**Description:** Fixed a rare issue where the kernel API `ib_free_cq()` could crash if a new IRQ or CQ work was submitted immediately after disabling the IRQ or canceling CQ work, potentially running concurrently with `destroy_cq`.
	Keywords: ib_free_cq, destroy_cq
	Discovered in Release: 23.10-5.1.4.0
	Fixed in Release: 23.10-6.1.6.1
4573786 / 4500815	**Description:** Fixed an issue that caused packet loss when enabling or disabling promiscuous mode on a network interface.
	Keywords: Promiscuous mode
	Discovered in Release: 23.10-5.1.4.0
	Fixed in Release: 23.10-6.1.6.1
4639994 / 4634995 / 4635609 / 4635610 / 4797618	**Description:** Fixed an issue where `mlx5dv_devx_umem_reg_ex()` would fail if `ib_umem_dmabuf_get_pinned()` was not defined.
	Keywords: UMEM_DMABUF_GET_PINNED, ib_umem_dmabuf_get_pinned
	Discovered in Release: 23.10-5.1.4.0
	Fixed in Release: 23.10-6.1.6.1
4648901 / 4606693	**Description:** Fixed a rare crash that could occur when a MAD completion queue (CQ) was destroyed while RDMA CM traffic was still active.
	Keywords: MAD CQ

Internal Reference Number	Description
	Discovered in Release: 23.10-5.1.4.0
	Fixed in Release: 23.10-6.1.6.1
4719792 / 4553499	**Description:** Fixed an issue where, on devices that do not support BlueFlame, allocation of a new Transport Domain (TD) could fail when attempting to allocate a dedicated UAR.
	Keywords: BlueFlame, Transport Domain (TD), UAR
	Discovered in Release: 23.10-5.1.4.0
	Fixed in Release: 23.10-6.1.6.1

Internal Reference Number	Description
4410029	**Description:** Fixed an issue where installing mlnx-ofa_kernel drivers on SLES 15 SP5 with kernel version 5.14.21-150500.55.68-default (and newer) failed due to weak-modules falling back to the original inbox modules. The failure was caused by a mismatch: the original build kernel (5.14.21-150500.53-default) did not include the mana_ib driver, so no dummy module was provided, while the newer kernel did include it. This mismatch led to weak-modules sanity check errors due to the presence of the inbox mana_ib driver.
	Keywords: mlnx-ofa_kernel, SLES 15 SP5
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4471811	**Description:** Resolved NVMe driver compilation issue on Linux kernel version 6.6.87.
	Keywords: NVMe driver
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4253229	**Description:** Fixed a race condition between the firmware syndrome report and driver initialization during boot.
	Keywords: Race condition
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4442965	**Description:** Fixed performance degradation on older kernel versions using RX cache, particularly on slower ARM CPUs with larger RX buffers. The issue was caused by the driver attempting to allocate new RX pages too quickly, leading to head-of-line blocking in the RX cache. The fix improves RX cache usage by triggering page allocation for a bulk of at least 2 WQEs, allowing the application more time to process packets and return buffers to the RX cache, thereby reducing blocking and enhancing performance.
	Keywords: Performance, kernel, Rx cache, page allocation

Internal Reference Number	Description
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4243800	**Description:** Resolved improper page deallocation handling issue present in some kernels.
	Keywords: Page deallocation
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4466255	**Description:** Fixed an issue where a kernel crash could occur if a device event arrives during the event subscription process.
	Keywords: DevX, event_fd
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4441119	**Description:** Fixed a crash caused by handling multiple CMA net events occurring in quick succession on the same CMA ID.
	Keywords: CMA
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4405723	**Description:** Fixed a potential deadlock that could occur during the handling of peer memory registration failures.
	Keywords: Deadlock, peer memory registration
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4340109	**Description:** Fixed a sysfs issue that occurred when accessing hardware counters from within a namespace.
	Keywords: sysfs
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4248125	**Description:** Fixed the UMR QP recovery flow to ensure proper functionality and prevent tasks from getting stuck in the kernel. Additionally, resolved a race condition in the ODP MR area that could lead to a CQE error in the UMR QP.
	Keywords: UMR QP recovery flow
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0
4235682	**Description:** Resolved corruption of SA MAD Congestion Control FIFO queue when all elements are canceled and a dequeue operation is attempted.
	Keywords: SA legacy congestion control mechanism
	Discovered in Release: 23.10-4.0.9.1

Internal Reference Number	Description
	Fixed in Release: 23.10-5.1.4.0

4409282	**Description:** Increased the size of the slow FDB table to prevent hitting the following error when switching to SwitchDev mode.
	`mlx5_core 0000:03:00.0: mlx5_cmd_out_err:835:(pid 24362): CREATE_FLOW_GROUP(0x933) op_mod(0x0) failed, status bad parameter(0x3), syndrome (0x4065f0), err(-22)`
	`mlx5_core 0000:03:00.0: E-Switch: Failed to create peer miss flow group err(-22)`

	Keywords: Slow FDB table
	Discovered in Release: 23.10-4.0.9.1
	Fixed in Release: 23.10-5.1.4.0

Internal Reference Number	Description
4192798	**Description:** Fixed the issue where the device failed to initialize after a write-combine test failure. The device now loads with Blueflame capabilities disabled instead.
	Keywords: write-combine; Blueflame; device initialization
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
4111625	**Description:** MLNX_OFED can now successfully be built with `add-kernel-support` flag over SLES15-SP5 kernel 5.14.21-150500.55.73.
	Keywords: SLES; Kernel; operating system; OS
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
4128400	**Description:** Removed the udev rule error message "/usr/lib/udev/rules.d/90-ib.rules:4 Only network interfaces can be renamed" from the log files. The udev rule included a line in a syntax that is no longer valid that triggered the mentioned error.
	Keywords: udev rule
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
4001035 / 4001038	**Description:** Fixed an issue that resulted in corrupt SMP MAD requests list when the sent list was accessed while the unregistered flow was running.
	Keywords: SMP MAD requests
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
3991835	**Description:** Fixed a stack overrun warning by reducing the size of the local on-stack array used for optimization by 192 bytes.
	Keywords: Kernel Stack

Internal Reference Number	Description
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
4004300	**Description:** Fixed an issue that prevented netdev queue value from being updated in mqprio param when switchdev mode was enabled and the netdev queue number was reset to 1.
	Keywords: PF TXQ mapping
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
4176804	**Description:** Fixed the receive queue cache size calculation to take into account the host page size.
	Keywords: Memory allocation
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
4027218	**Description:** Fixed the packet inspection parsing to avoid data corruption when GRE offload was turned on by parsing the outer header as UDP and not as TCP.
	Keywords: UDP, TCP, GRE offload
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
4001551	**Description:** Fixed a CT entry update failure that was caused because of a firmware limitation, the old modify header context was not freed and had leaks.
	Keywords: CT entry update
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
4012710	**Description:** Increased the `MLX5E_TC_MAX_INT_PORT_NUM` value to 32 to avoid cases of rules not being offloaded.
	Keywords: Rules offloaded
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1
4014362	**Description:** Fixed an issue that prevented the "hash" `lag_port_select_mode` from working properly with ConnectX-7 adapter cards on some old OSs.
	Keywords: LAG
	Discovered in Release: 23.10-3.2.2.0
	Fixed in Release: 23.10-4.0.9.1

Internal Reference Number	Description
3932946	**Description:** Fixed the setting of ATS for DMABUF MRs that caused some MRs to miss the ATS enablement. Lack of ATS enablement on DMABUF MRs results in slower performance when using these MRs.
	Keywords: ATS, DMABUF, MRs
	Discovered in Release: 23.10-2.1.3.1
	Fixed in Release: 23.10-3.2.2.0
3894403	**Description:** On rare occasions, rdmacm applications could not find the device upon creating new RDMA devices, as the CMA driver lost some of the devices due to an overflow issue.
	Keywords: rdmacm
	Discovered in Release: 23.10-2.1.3.1
	Fixed in Release: 23.10-3.2.2.0
3807155	**Description:** Fixed an issue that could have caused memory corruption when running XDP traffic.
	Keywords: tc_wrap tool, VLAN
	Discovered in Release: 23.10-2.1.3.1
	Fixed in Release: 23.10-3.2.2.0
3848999	**Description:** Fixed an issue that prevented the tc_wrap tool from properly working when VLAN is configured as the tool wrongly handled the library function return value.
	Keywords: tc_wrap tool, VLAN
	Discovered in Release: 23.10-2.1.3.1
	Fixed in Release: 23.10-3.2.2.0
3822520	**Description:** Fixed an issue that caused a deadlock in the flow of disabling the LAG when changing eswitch mode from switchdev to legacy when a LAG bond existed on the machine.
	Keywords: MR cache cleanup
	Discovered in Release: 23.10-2.1.3.1
	Fixed in Release: 23.10-3.2.2.0
3822520	**Description:** Fixed an issue related to the comparison process between the SW steering and FW steering modes to avoid kernel crashes incidences.
	Keywords: MR cache cleanup
	Discovered in Release: 23.10-2.1.3.1
	Fixed in Release: 23.10-3.2.2.0
3947195	**Description:** Fixed an issue related to the driver's internal MR cache cleanup that caused high memory consumption on the host.
	Keywords: MR cache cleanup
	Discovered in Release: 23.10-2.1.3.1
	Fixed in Release: 23.10-3.2.2.0

Internal Reference Number	Description
3729466	**Description:** Resolved a discalculation issue where more Q-counters were freed than allocated when moving to switchdev mode.
	Keywords: Q-counters, switchdev
	Discovered in Release: 23.10-1.1.9.0
	Fixed in Release: 23.10-2.1.3.1
3727822	**Description:** Fixed an issue that allowed concurrent creation of encap entries, and could potentially cause double free vulnerabilities.
	Keywords: encap entries, double free
	Discovered in Release: 23.10-1.1.9.0
	Fixed in Release: 23.10-2.1.3.1
3728381	**Description:** Fixed an issue that exposed debugfs entries for non supported RoCE general parameters, such as rtt_resp_dscp.
	Keywords: debugfs, RoCE
	Discovered in Release: 23.10-1.1.9.0
	Fixed in Release: 23.10-2.1.3.1
3710957	**Description:** Fixed an issue that triggered an error message by updating the rule actions STE apply flow. Following the update, the flow checks if the rule domain is different from the ASO CT action domain when applying the ASO CT action.
	Keywords: Software Steering
	Discovered in Release: 23.10-1.1.9.0
	Fixed in Release: 23.10-2.1.3.1

Internal Reference Number	Description
3663363	**Description:** Fixed an issue where an error was triggered in case devlink reload was attempted when there were allocated subfunctions.
	Keywords: devlink reload, allocated subfunctions
	Discovered in Release: 23.10-0.5.5.0
	Fixed in Release: 23.10-1.1.9.0
3660998	**Description:** Resolved an issue on ConnectX-4 Lx, where the VF state was not configured correctly following the activation of SR-IOV.
	Keywords: ConnectX-4 Lx, VF state
	Discovered in Release: 23.10-0.5.5.0
	Fixed in Release: 23.10-1.1.9.0
3653417	**Description:** Fixed an issue where changing the steering mode to firmware steering was unsupported for policy IPsec rules.
	Keywords: Firmware steering
	Discovered in Release: 23.10-0.5.5.0

Internal Reference Number	Description
	Fixed in Release: 23.10-1.1.9.0

Internal Reference Number	Description
3613899	**Description:** Fixed an issue where the srp_daemon service was enabled by default. Starting from this release, the srp_daemon service is disabled by default, and can be enabled after MLNX_OFED installation using the `systemctl enable srp_daemon` command.
	Keywords: srp_daemon
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3549785	**Description:** Fixed an issue where the NVMe and mlx5_core drivers failed during BFB installation. As a result, Anolis OS could not be installed on the SSD and the mlxfwreset command did not work during Anolis BFB installation.
	Keywords: NVMe, mlx5_core, Anolis OS
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3602955	**Description:** Fixed an issue that occurred when a VF was set to get allmulti traffic. The issue caused the steering rules to send the multicast traffic received by the NIC back to the uplink.
	Keywords: VF, allmulti traffic
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3553766	**Description:** Fixed an issue where the `enable_remote_dev_reset` Devlink parameter was not supported on kernel versions below v5.10.
	Keywords: Devlink parameter
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3546694	**Description:** Fixed an issue where MAC address configuration for PFs could fail if SR-IOV was enabled at the same time.
	Keywords: PF, MAC address, SR-IOV
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3538018	**Description:** Fixed an issue where firmware sync reset (with the '`mlxfwreset -d <device> -l 3 r --sync 1`' command) could fail on a system configured for hotplug on the PCIe slot on which the mlx5 card was mounted.
	Keywords: Firmware sync reset, mlx5 card

Internal Reference Number	Description
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3587834	**Description:** Fixed an issue where the `enable_remote_dev_reset` Devlink parameter was not supported on kernel versions below v5.10.
	Keywords: Devlink parameter
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3576351	**Description:** Resolved a warning that was triggered when starting the openibd service, which pertained to an unidentified 'ExecRestart' value within the 'Service' section.
	Keywords: openibd, warning
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3565980	**Description:** Fixed an issue where deb-based (Ubuntu and Debian) versions of MLNX_OFED did not install the rshim package, and the software needed to access the BlueField guest system from the host on a BlueField-type installation by default. Note that this was only needed (and enabled) on the host side. This issue did not occur on RPM-based systems (RHEL, SLES, Oracle, etc.).
	Keywords: Ubuntu, Debian, rshim package
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3557482	**Description:** Fixed an issue where the 'mlnx_tune -l' list of supported operating systems did not include several operating systems that were actually supported, such as RHEL8.6 and Ubuntu 22.04.
	Keywords: mlnx_tune -l
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3549684	**Description:** Fixed a signature-related issue that occurred when installing DOCA on SLES15SP4 using the repository.
	Keywords: DOCA, SLES15SP4
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3380263	**Description:** Fixed an issue where users who attempted to use OFED with Device ID NVD0000000033, had to install the firmware manually.
	Keywords: Device ID NVD0000000033
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0
3228788	**Description:** Fixed an issue where running rx-tls-offload over Korg6.0 as its TLS module did not work properly.
	Keywords: NetDev, TLS

Internal Reference Number	Description
	Discovered in Release: 23.07-0.5.0.0
	Fixed in Release: 23.10-0.5.5.0

Internal Reference Number	Description
3546304	**Description:** Resolved the kernel crash resulting from sysfs calls to profiles lacking TC (Traffic Control) support.
	Keywords: sysfs calls, Trafic Control
	Discovered in Release: 23.04-0.5.3.3
	Fixed in Release: 23.07-0.5.0.0
3531986	**Description:** Fixed an issue that prevented OS booting following an installation of the EN and RoCE drivers.
	Keywords: OS booting, EN, RoCE
	Discovered in Release: 23.04-0.5.3.3
	Fixed in Release: 23.07-0.5.0.0
3489233	**Description:** Fixed an issue in SLES 15 SP4 where the openibd service failed to start automatically after system boot.
	Keywords: SLES 15 SP4,openibd, system boot
	Discovered in Release: 23.04-0.5.3.3
	Fixed in Release: 23.07-0.5.0.0
3431430	**Description:** Fixed an issue that prevented the installation of OFED on RHEL systems using a non-default Python version.
	Keywords: Installation, RHEL, Python
	Discovered in Release: 5.9-0.5.6
	Fixed in Release: 23.07-0.5.0.0
3422823	**Description:** Fixed an OFED installation issue on BCLinux 21.10 that occurred when using the "--add-kernel-support" installation flag.
	Keywords: Installation, BCLinux 21.10, "--add-kernel-support"
	Discovered in Release: 5.9-0.5.6
	Fixed in Release: 23.07-0.5.0.0
3264588	**Description:** Resolved a problem where the system boot process would hang when more than two Network Interface Cards were installed.
	Keywords: System boot, Network Interface Cards
	Discovered in Release: 5.7-1.0.2.0
	Fixed in Release: 23.07-0.5.0.0
3499136	**Description:** Fixed an issue where the sysfs PHY counters displayed outdated information.

Internal Reference Number	Description
	Keywords: sysfs PHY counters
	Discovered in Release: 23.04
	Fixed in Release: 23.07-0.5.0.0

Internal Reference Number	Description
2883451	**Description:** Installing mlnx_tune on Python3 did not work properly. mlnx_tune now supports Python3 in addition to Python2.
	Keywords: Installation, mlnx_tune, Python3
	Discovered in Release: 5.5-1.0.3.2
	Fixed in Release: 23.04-0.5.3.3
3219842	**Description:** When creating a bond interface for all ports on a ConnectX-7 4-port HCA, the wrong bond name appeared in ibdev2netdev.
	Keywords: RDMA, Bond Name, ibdev2netdev, ConnectX-7
	Discovered in Release: 5.8-1.0.1.1
	Fixed in Release: 23.04-0.5.3.3
3333919	**Description:** Changing traffic class via the sysfs while modifying QPs in parallel causes a deadlock.
	Keywords: RDMA, TC, Sysfs, QP
	Discovered in Release: 5.0-2.1.8.0
	Fixed in Release: 23.04-0.5.3.3
3406019	**Description:** Due to a bug in the emulation layer, performance degradation might be experienced when running GPUDirect over Virtual Functions.
	Keywords: RDMA, GPUDirect, performance, VF
	Discovered in Release: 5.9-0.5.6.0
	Fixed in Release: 23.04-0.5.3.3
3233799	**Description:** debugfs directories cannot be created for representors and sub-functions, thus the log might show error warning for either of the scenarios.
	Keywords: NetDev, debugfs, SF, logging
	Discovered in Release: 5.8-1.0.1.1
	Fixed in Release: 23.04-0.5.3.3
1892663/1800633/2883451	**Description:** mlnx_tune script does not support Python3 interpreter.
	Keywords: mlnx_tune, Python3
	Discovered in Release: 4.7-1.0.0.1
	Fixed in Release: 23.04-0.5.3.3

Internal Reference Number	Description
3340542	**Description:** The verion number for perftest was not-standard resulting in some distribution packages recieving a higher version number than the OFED version for no good reason. Changed the naming of perftest to MAJOR.MINOR.PATCH.
	Keywords: Installation, perftest
	Fixed in Release: 23.04-0.5.3.3
3428775	**Description:** knem did not fully support RHEL8.7 and newer releases.
	Keywords: Installation, knem, RHEL
	Discovered in Release: 5.8-1.0.1.1
	Fixed in Release: 23.04-0.5.3.3
3431430	**Description:** Installing MLNX_OFED on a RHEL system that uses a non-default version of Python (e.g., Python3.9 on RHEL8.6, where the default is 3.6) may fail with an error that mlnx-tools is missing a dependency on 'python(abi)'. mlnx-tools includes a single script, mlnx_qos, that depends on a specific version of python. In such a case, after the fix, it may fail to run with such a non-standard version of Python.
	Keywords: Installation, Python, RHEL, mlnx-tools
	Discovered in Release: 5.9-0.5.6.0
	Fixed in Release: 23.04-0.5.3.3

5.2 User Manual Revision History

Release	Date	Description
23.10-6	January 2026	No changes to the User Manual.
23.10	November 2023	• Added IPsec Full Offload • Added a note to NVME-oF - NVM Express over Fabrics.
23.07	September 2023	No changes to the User Manual.
	August 2023	• Updated "Interrupt Request (IRQ) Naming" under Ethernet Interface. • Removed the "IPsec Crypto Offload" section from the Ethernet Network chapter. • Removed all "Connected Mode" indications, and the "IPoIB Mode Setting" section from IP over InfiniBand (IPoIB). • Removed Running OpenSM, and updated Running OpenSM as Daemon under NVIDIA SM.
23.04	May 2023	• Removed "Installing NEO-Host Using mlnxofedinstall Script" subsection from the Installing MLNX_OFED section • Added note under "Enrolling NVIDIA's x.509 Public Key On your Systems" subsection in the UEFI Secure Boot section • Updated the NVIDIA SM (OpenSM) section
	Jun 2023	• Added an important note to IPsec Crypto Offload
5.7	August 2022	• Added Out of Order (OOO) under RoCE section • Added section describing Dumping Steering Info

Release	Date	Description
5.6	April 2022	• Added OpenFlow Meters section • Added Installation on Community Operating Systems section
5.5	December 2021	• Added "Forward to Multiple Destinations" section • Added "Open vSwitch Metering" section • Added "Multiport eSwitch Mode" section • Added "Bridge Offload" section • Added "TC Configuration for ConnectX-6 Dx and Above" subsection
5.4-3	October 2021	• Removed LASH Routing Algorithm
5.4-2	August 2021	• Added CT CT NAT section • Added Representor Metering section • Updated VF Metering section • Removed sysfs VXLAN portion of the Enabling VXLAN Hardware Stateless Offloads section
5.4	June 2021	• Updated SR-IOV Live Migration
5.3-1	March 31,2021	• Added PTP Cyc2time Hardware Translation Offload section • Added Connection Tracking Performance Tuning section • Added VF Metering section • Added note under OVS-DPDK Hardware Offloads section • Updated Persistent Naming section • Updated command under Connection Tracking Offload section • Added sFlow description under OVS-DPDK Hardware Offloads section
5.2	February 14, 2021	• Added Setting up MLNX_OFED YUM Repository Using --add-kernel-support section • Added Setting up MLNX_OFED apt-get Repository Using --add-kernel-support section
	January 19, 2021	Added OpenSSL with kTLS Offload section
	January 4, 2021	• Added Offloaded Traffic Sniffer section • Added Tx Port Time-Stamping section • Added VLAN Push/Pop section • Added sFLOW section • Added E2E Cache section • Added Geneve Encapsulation/Decapsulation section • Added Parallel Offloads section • Updated SR-IOV VF LAG section • Removed Installing MLNX_OFED on Innova™ IPsec Adapter Cards section • Removed Updating Firmware and FPGA Image on Innova IPsec Cards section
5.1-2	September 17, 2020	Added Packet Pacing for Hairpin Queues section
5.1	July 28, 2020	Updated the content of the entire document following the removal of support for ConnectX-3, ConnectX-3 Pro and Connect-IB adapter cards, as well as the deprecation of RDMA experimental verbs library (mlnx_lib)
		Added SR-IOV Live Migration section
		Added SR-IOV VF LAG section
5.0-2	April 23, 2020	Added Interrupt Request (IRQ) Naming section
	April 6, 2020	Added Kernel Transport Layer Security (kTLS) Offloads section
5.0	March 3, 2020	• Added IPSec Crypto Offload section • Added OVS-DPDK Hardware Offloads section • Updated OVS Hardware Offloads Configuration section

Release	Date	Description
4.7	December 29, 2019	• Added Configuring Uplink Representor Mode section
	December 13, 2019	• Added Performance Tuning Based on Traffic Patterns section • Added "num_of_groups" entry to table mlx5_core Module Parameters • Added Mediated Devices section
	September 29, 2019	• Updated Additional Installation Procedures section
4.6	May 13, 2019	• ethtool section updates: Added description of -f flashing option to Ethtool Supported Options table

	April 30, 2019	• ethtool section updates:
		  • Updated the description of ethtool -s eth<x> advertise <N> autoneg on counter under Ethtool
		  • Added the following counters under Ethtool:
		  • ethtool --show-fec eth<x>
		  • ethtool --set-fec eth<x> encoding auto|off|rs|baser
		• Added Devlink Parameters section
		• Added Limit Bandwidth per Group of VFs section
		• Added Disabling RoCE section
		• Added RDMA-CM QP Timeout Control section
		• Added RDMA-CM Application Managed QP section

4.5	December 19, 2018	• Reorganized Chapter 2, “Installation”: Consolidated the separate installation procedures under Installing NVIDIA OFED and Additional Installation Procedures • Added Installing NEO-Host Using mlnxofedinstall Script
	November 29, 2018	Added the following sections: • Local Loopback Disable • Offsweep Balancing

6 Legal Notices and 3rd Party Licenses

The following are the drivers' software, tools and HCA firmware legal notices and 3rd party licenses.

Product	Version	Legal Notices and 3rd Party Licenses
MLNX_OFED	23.10-6	• License • 3rd Part Notice
Firmware	xx.39.51xx	• HCA Firmware EULA • 3rd Party Unify Notice • License
MFT	4.26.1-35	• License • 3rd Party Notice • 3rd Party Unify Notice
Clusterkit	1.11	• License • 3rd Party Notice
DPCP	1.1.43	• License • 3rd Party Notice
VMA	9.8.40	• 3rd Party Unify Notice • 3rd Party Notice
XLIO	3.20.8	• License • 3rd Party Unify Notice
HCOLL	4.8	• License • 3rd Party Notice
SHARP	3.5.2	• License • 3rd Party Notice
ibutils2	2.15	• License • 3rd Party Notice
OpenSM	5.17.2	• 3rd Party Unify Notice • 3rd Party Notice
mpitests	3.2.21	• License • 3rd Party Notice

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