NVIDIA Firmware Tools (MFT) Documentation v4.34.0

Table of contents

Overview

The NVIDIA® Firmware Tools (MFT) package is a set of firmware management and debug tools for NVIDIA devices. The document describes MFT features, tools content and configuration.

The documentation here relates to:

- Release Notes
- User Manual

Intended Audience

This manual is intended for system administrators responsible for managing and debugging firmware for NVIDIA devices.

See also Document Conventions and Related Documents.

Software Download

To download product software, please refer to the Firmware Tools (MFT) product page.

Document Revision History

A list of the changes made to the user manual are provided in User Manual Revision History.

Release Notes

These are the release notes for NVIDIA® Firmware Tools (MFT). MFT supports the following operating systems: Linux, Windows, VMware ESXi and FreeBSD. Please see the supported platform table for further details.

The tools functionality is identical in all operating systems unless otherwise noted.

Release Notes Update History

Revision	Date	Description
4.34.0-145	November 10, 2025	Initial release of this Release Notes version. This version introduces Bug Fixes.

General Information

Package Tools

The following is a list of the available tools in the package, together with a brief description of each tool. The tools apply to single switch systems or adapter cards.

> **Note**
>
> Please note that running MFT tools in parallel or simultaneously on the same device is not supported and might cause unexpected behavior.

The MFT tools do not provide cluster wide functionality.

Category	Tool	Description	Operating System
MST Service	mst	• Lists the available mst devices • Start/stop the register access driver for Linux and VMware ESXi OSs.	All
Firmware Update and Configuration	mlxburn	This tool provides the following functions: • Generating a standard or customized firmware image for burning in .bin format • Burning an image to the Flash attached to an HCA or switch device • Querying the firmware version loaded on a device • Displaying the Vital Product Data (VPD) of a network adapter	All
	flint	This tool burns a firmware binary image or an expansion ROM image to the Flash of a network adapter/ switch device. It includes query functions to the burnt firmware image and to the binary image file.	All
	mlxconfig	Allows the user to change some of the device configurations without having to create and burn a new firmware.	All
	mlxfwmanager	The mlxfwmanager is a firmware update and query utility. It provides a simple 'single click' firmware update functionality. **Note:** The same tool with embedded firmware binaries is released separately and is named mlxup.	All
	mlxarchive	The mlxarchive tool allows the user to create a file with the mfa2 extension. The new file contains several binary files of a given firmware for different adapter cards.	Linux Windows FreeBSD
	mlxphyburn	A tool for burning externally managed PHY	Linux
	mlx_fpga	A tool for burning and debugging devices with FPGA. It allows the user to burn their own hardware code on an FPGA integrated with HCA board. It also provides the user	Linux

Category	Tool	Description	Operating System
		with read/write registers in the QDR memory of the FPGA.	
	cpldupdate	A tool for programing on board CPLDs for NVIDIA devices for the OEM packages only.	Linux
Debug and Diagnostics Utilities	itrace	Extracts and prints trace messages generated by the firmware of a ConnectX-3 adapter cards.	All
	fwtrace	Extracts and prints trace messages generated by the firmware of 5th generation devices	Linux Windows FreeBSD
	mlxtrace	Dumps trace messages generated by the device hardware.	All
	mlxdump	Dumps device internal configuration registers. The dump file can be used by the Support team for hardware troubleshooting.	All
	mlxmcg	Displays the current multicast groups and flow steering rules configured in the device. Target users: Developers of Flow Steering aware applications.	All
	wqdump	Dumps the current QP contexts and Work Queues of ConnectX family network adapter cards and Connect-IB adapter cards.	All
	i2c	Generates an i2c transaction using an mtusb usb to i2c adapter or using the device internal i2c compatible master	Linux Windows FreeBSD
	mlxi2c	Scans the i2c bus Routes the i2c bus of an externally managed InfiniscaleIV/SwitchX system to connect to the switch silicon.	Linux Windows
	mget_temp	Reads the hardware temperature from NVIDIA devices internal sensors and prints the reading in Celsius degrees.	All

Category	Tool	Description	Operating System
	pckt_drop	Corrupts the next transmitted packet from the ConnectX family network adapter cards and Connect-IB adapter cards.	All
	mlxuptime	Prints NVIDIA devices' up time and measured/configured core clock frequency	All
	mlxfwreset	Loads the firmware after firmware update on ISFU capable devices. (5th generation devices)	Linux Windows FreeBSD
	mlxmdio	Reads/writes MDIO registers (Clause 45) on boards with externally managed PHY	All
	mlxreg	Exposes supported access registers, and allows users to obtain information regarding the registers fields and attributes, and to set and get data with specific register.	All
	mstdump	Dumps device internal configuration data.	All
	mcra	Reads/writes a single word from/to a device configuration register space	All
	mlxcables	Reads/writes NVIDIA cable registers and queries the cables info	All
	mlxlink	Displays and configures port related data at the physical layer.	All
	mlxvpd	Reads PCI device VPD	All
	mlxprivhost	Enables the user to restrict the hosts from configuring the NIC.	Linux
	resourcedump	Extracts and prints data segments generated by the firmware.	Linux Windows
	resourceparse	Parses and prints data segments content.	Linux Windows
	stedump	A packet simulator for host NIC steering solutions.	Linux

Detailed installation instructions along with complete descriptions of the various tools in the package can be found in the Firmware Tools User Manual.

Software Dependencies

Software Package	Required Version
Linux	
Kernel sources	Machine's kernel version
DOCA-Host [1, 2]	1.5.0 or later
Perl	5.24 or later
Python[3]	3.6 or later
lsusb[4]	
rpmbuild	
xz[5]	
Windows	
NVIDIA WinOF[6]	3.0.0 or later
VMware ESXi	
Python	3.6 or later

Notes:

1. DOCA-Host can be downloaded from http://www.openfabrics.org. Note that installing DOCA-Host is *not* required if you wish to install MFT without In-Band capabilities.

2. For the 'mst ib add' command to run, one of the DOCA-Host packages "ibutils" or "ibutils2" or "infiniband-diags" should be installed and available in the PATH. (For details on DOCA-Host installation, visit https://docs.nvidia.com/networking/category/mlnxofedib.)

3. Required for the mlxmcg tool only.

4. Required for the mtusb device usage.

5. For creating UPMF (update package for NVIDIA firmware).

6. WinOF is required only for In-Band access. The package can be downloaded fromthe WinOF page..

7. Python 2.x is now end-of-life and no longer supported by MFT. To use the latest and up-to-date MFT tools, we recommend you use Python 3.x.

Dependencies

3rd Part MFT Dependencies

Component name	Component version name	Home Page	License names
JSON-CPP	0.6.0	https://github.com/open-source-parsers/jsoncpp	MIT License
OpenSSL	1.1.1l	http://www.openssl.org/	The Open SSL License
Perl	v5.32.0	http://www.perl.org/	Artistic License 1.0 (Perl)
SQLite	3.33.0	http://sqlite.org/	Public Domain
XZ Utils	5.0.4	http://tukaani.org/xz/	Public Domain
Curl	7.79.1	https://curl.se/	curl License
Iniparser	4.1	http://ndevilla.free.fr/iniparser	MIT License
Libexpat	2.4.1	http://www.libexpat.org/	MIT License
libxml2	2.9.10	http://www.xmlsoft.org/	MIT License
Zlib	1.2.11	http://www.zlib.net/	zlib License

Supported Operating Systems and Platforms

MFT is supported on the following platforms:

Table Legend:

+ (Green)	Supported and tested
** (Orange)	Supported but not tested

*** Blue	Partially tested

Supported Operating Systems and Platforms

OS	Platform	Status
RH/ CentOS 8.2	x86_64	+
RH/ CentOS 8.2	PPC64LE	+
RH/ CentOS 8.4	PPC64LE	**
RH/ CentOS 8.4	x86_64	+
RH/ CentOS 8.6	x86_64	+
RH/ CentOS 8.6	PPC64LE	+
RH/ CentOS 8.8	x86_64	+
RH/ CentOS 8.8	PPC64LE	**
RH/ CentOS 8.8	ARM64	+
RH/ CentOS 8.9	x86_64	+
RH/ CentOS 8.9	PPC64LE	+
RH/ CentOS 8.9	ARM64	+
RH/ Rocky 9.0	x86_64	+
RH/ Rocky 9.0	PPC64LE	+
RH/ Rocky 9.1	x86_64	+
RH/ Rocky 9.1	PPC64LE	+
RH/ Rocky 9.2	x86_64	+
RH/ Rocky 9.2	PPC64LE	+
RH/ Rocky 9.2	ARM64	+
RH/ Rocky 9.3	x86_64	+
RH/ Rocky 9.3	PPC64LE	+
RH/ Rocky 9.3	ARM64	+
RH/ Rocky 9.4	x86_64	+
RH/ Rocky 9.4	PPC64LE	+
RH/ Rocky 9.4	ARM64	+

OS	Platform	Status
RH/ Rocky 9.5	x86_64	+
RH/ Rocky 9.5	PPC64LE	+
RH/ Rocky 9.5	ARM64	+
RH/ Rocky 9.6	x86_64	+
RH/ Rocky 9.6	PPC64LE	+
RH/ Rocky 9.6	ARM64	+
RH/ Rocky 10	x86_64	+
RH/ Rocky 10	PPC64LE	+
RH/ Rocky 10	ARM64	+
Centos Stream v8 - Community	x86_64	**
Centos Stream v8 - Community	PPC64LE	**
Centos Stream v9 - Community	x86_64	+
Centos Stream v9 - Community	ARM64	+
Centos Stream v9 - Community	PPC64LE	**
OEL 8.4	x86_64	+
OEL 8.6	x86_64	+
OEL 8.7	x86_64	+
OEL 8.8	x86_64	+
OEL 9.0	x86_64	+
OEL 9.1	x86_64	+
OEL 9.2	x86_64	+
OEL 9.5	x86_64	+
Fedora 32 - Community	x86_64	+
Fedora 35 - Community	x86_64	**
Sles15 SP2	x86_64	+
Sles15 SP2	PPC64LE	+
Sles15 SP3	PPC64LE	+
Sles15 SP3	x86_64	+

OS	Platform	Status
Sles15 SP4	PPC64LE	+
Sles15 SP4	x86_64	**
Sles15 SP5	PPC64LE	+
Sles15 SP5	x86_64	+
Sles15 SP6	PPC64LE	+
Sles15 SP6	x86_64	+
Sles15 SP7	x86_64	+
Sles15 SP7	PPC64LE	+
Sles15 SP7	ARM64	+
EulerOS V2.0 SP9 - Community	x86_64	**
EulerOS V2.0 SP10 - Community	x86_64	+
EulerOS V2.0 SP11	x86_64	+
EulerOS V2.0 SP12	x86_64	+
EulerOS V2.0 SP12	ARM64	+
OpenEuler 20.3 SP1 - Community	x86_64	**
OpenEuler 20.3 SP3	x86_64	+
OpenEuler 22.3 LTS	x86_64	+
OpenEuler 22.3 SP1	x86_64	+
Ubuntu 20.04	PPC64LE	+
Ubuntu 20.04	ARM64	+
Ubuntu 20.04	x86_64	+
Ubuntu 22.04	x86_64	+
Ubuntu 23.10	x86_64	+
Ubuntu 24.04	x86_64	+
Ubuntu 24.04	ARM64	+
Ubuntu 24.04	PPC64LE	**
Ubuntu 25.04	x86_64	+
BCLinux 21.10 SP2	x86_64	+

OS	Platform	Status
BCLinux 21.10 SP2	ARM64	**
BCLinux 22.10 SP2	x86_64	+
BCLinux 22.10 SP2	ARM64	+
Debian 10.8	x86_64	+
Debian 10.9	x86_64	+
Debian 10.13	x86_64	+
Debian 10.13	ARM64	+
Debian 11.3	ARM64	+
Debian 11.3	x86_64	+
Debian 12.1	x86_64	+
Debian 12.1	ARM64	+
Debian 12.5	x86_64	+
Debian 12.5	ARM64	+
Debian 13.0	x86_64	+
Debian 13.0	ARM64	+
Citrix server 8.2	x86_64	+
Anolis 8.4 - Community	x86_64	**
Anolis 8.6	x86_64	+
Anolis 8.6	ARM64	+
Korg 6.8	x86_64	+
Korg 6.8	ARM64	+
TKLinux 3.3	x86_64	+
TKLinux 3.3	ARM64	**
OpenSUSE 15.3 - Community	x86_64	**
Photon 3.0 - Community	x86_64	**
CTYunOS2	x86_64	+
CTYunOS2	ppc64le	**
CTYunOS3	x86_64	**

OS	Platform	Status
CTYunOS3	ARM64	+
Mariner 2.0	x86_64	+
Alma 8.5	x86_64	**
KylinOS v10 SP2	x86_64	+
KylinOS v10 SP3	x86_64	+
KylinOS v10 SP3	ARM64	+
Allinux 3.2	x86_64	+
Allinux 3.2	ppc64le	+
DriveOS 6.0.5.0	x86_64	+
DriveOS 6.0.5.0	ARM64	+
UOS v20 1040d -Community	x86_64	+
UOS v20 1060a	x86_64	+
UOS v20 1060a	ARM64	**
UOS v20 1060e	x86_64	+
UOS v20 1060e	ARM64	**
UOS v20 1021e	x86_64	+
UOS v20 1021e	ARM64	**
Windows Server 2016	64 Bit	+
Windows Server 2019	64 Bit	+
Windows Server 2022	64 Bit	+
Windows Server 2025	x86_64	+
Windows Server AH2023	64 Bit	+
Windows Server AH2022	64 Bit	+
Windows Server AH2021	64 Bit	+
Windows Server AH2020	64 Bit	+
Windows 10 21H2	64 Bit	+
Windows 10 22H2	64 Bit	+
Windows 10 1809	64 Bit	+

OS	Platform	Status
Windows 11 21H2	64 Bit	+
Windows 11 22H2	64 Bit	+
Windows 11 23H2	64 Bit	+
Windows 11 24H2	64 Bit	+
Windows 11 25H2	64 Bit	+
WinPE 4.0	32 Bit	+
WinPE 4.0	64 Bit	+
WinPE 5.0	32 Bit	+
WinPE 5.0	64 Bit	+
WinPE 5.1	32 Bit	+
WinPE 5.1	64 Bit	+
WinPE 10	32 Bit	+
WinPE 10	64 Bit	+
VMware ESXi 8.0 Native (Vsphere 2022)	64 Bit	+
VMware ESXi 8.0 Native (Vsphere 2022)	ARM	+
VMware ESXi 8.0 u1 Native (Vsphere 2022)	64 Bit	+
VMware ESXi 8.0 u1 Native (Vsphere 2022)	ARM64	+
VMware ESXi 8.0 u2 Native (Vsphere 2022)	x86_64	+
VMware ESXi 8.0 u2 Native (Vsphere 2022)	ARM64	+
VMware ESXi 8.0 u3 Native	64 Bit	+
VMware ESXi 8.0 u3 Native	ARM64	+
VMware ESXi 9.0 Native	64 Bit	+
VMware ESXi 9.0 Native	ARM64	+
FreeBSD 14.0-STABLE	x86_64	+
FreeBSD 15-CURRENT	x86_64	+
MLNX-OS 3.12.1000	64 Bit	+
SONiC 202311	64 Bit	**
Cumulus 5.9	64 Bit	+

OS	Platform	Status
NV-OS 25.02.1000	64 Bit	**
DVS 4.7.1000	64 Bit	+
FreeBSD 14.0-STABLE	ARM64	+
FreeBSD 15.0-CURRENT	ARM64	+
Korg6.10	X86_64	+
Korg6.11	X86_64	+
Mariner 3.0	X86_64	+
OpenEuler 22.03-SP3	X86_64	+
OpenEuler 22.03.LTS	X86_64	+
VMware ESXi 9.0 Native	x86	+
Xenserver 8.2	X86_64	+

Supported Flash Types

MFT supports the following Flash types.

Vendor	Flash Family	Tested P/N
Winbond	W25QxxBV	W25Q32FVSSIG
		W25Q32FVSSIGS
		W25Q32FVSSIGT
		W25Q128FVSIGS
	W25Qxxx	W25Q256JVBIMT
		W25Q128JVSIQ
Macronicx	MX25L16xxx	MX25L12845GM2I-08G
	MX25Lxxx	MX25L25645GXDI-08G
Micron	N25Q0xxx	MT25QL128ABA1ESE-0SIT
ISSI	IS25LPxxx	IS25LP128-JBLE SPA# U1323A
	IS25WPxxx	IS25WP256E-RHLE

Vendor	Flash Family	Tested P/N
Cypress	S25FL256L	S25FL256LDPBHV023
		S25FL128SAGMFVG00
Gigadevice		GD25LB256EBFRY
		GD25B256DFIGR

Supported NVIDIA Devices

The supported NVIDIA devices are listed in the following table:

IC Group	IC Device
Group II/5th Generation	• Adapter Cards: • ◦ NVIDIA ConnectX-8 ◦ ConnectX-8 - PCIe Switch ◦ NVIDIA BlueField-3 ◦ NVIDIA BlueField-2 ◦ NVIDIA ConnectX-7 ◦ ConnectX-7 - PCIe Switch ◦ NVIDIA ConnectX-6 Lx ◦ NVIDIA ConnectX-6 Dx ◦ NVIDIA ConnectX-6 • Switch Systems: ◦ NVIDIA Quantum-3 ◦ NVIDIA Quantum-2 ◦ NVIDIA Quantum ◦ NVIDIA Spectrum-4 ◦ NVIDIA Spectrum-3 ◦ NVIDIA Spectrum-2 ◦ NVIDIA Switch-IB 2 ◦ NVIDIA Switch-IB • GPU ◦ NVIDIA Blackwell • Retimer ◦ Arcus-E

Supported Adapter Cards Firmware Versions

The following NVIDIA® network adapter cards are supported:

Adapter Card	Bundled Firmware Version
ConnectX-8	40.47.1026
BlueField®-3	32.47.1026
BlueField®-2	24.47.1026
ConnectX-7	28.47.1026
ConnectX-6 Lx	26.47.1026
ConnectX-6 Dx	22.47.1026

To download the firmware binaries, please visit Firmware Downloads.

Supported Switch Systems Software

The following are the Supported Switch Systems Software.

Switch Software	Version
MLNX-OS	3.12.1000
SONIC	202311
Cumulus	5.9
DVS	4.7.1000
NV_OS	25.02.1000

Changes and New Features

Component/ Tool	Description	Operating System
Rev. 4.34.0-145		
nvredfish	Added support for nvredfish tool.	All
mdevices_info	Added `--enable-vfio` configure flag to make VFIO library optional.	All
mlxconfig	Added support for `--with_default` flag for mlxconfig tool.	All
	Added support for `--host_id` and `pf_index` flags in mlxconfig tool.	All
mlxlink	Added support for `--rx_modulation` and `--tx_modultion` flags to the test mode flow. Modulation status is shown in mlxlink output when in test mode.	All
mlxreg	Exposed the syndrome in mlxreg in external MFT package.	All
mlxdpa	Added commands for creating, signing, and removing a single application. Also, added support for `keep_sig` flag for container creation.	All

Customer Affecting Changes

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.

Planned for Version	Description
N/A	N/A

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
4.34.0	Starting with the October 2025 firmware release, and for all subsequent versions, compatibility with the older MFT releases (4.31.0-149 and 4.30.0-139) is no longer supported.

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.

Planned for Version	Description
N/A	N/A

Discontinued Features

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

N/A

Declared Unsupported Features

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

N/A

MFT Bug Fixes in this Version

For a list of old Bug Fixes, please see MFT Bug Fixes History.

Internal Ref.	Description
4564528	**Description:** In `mlx_fwsfx_gen`, r emoved dependency of /dev/null due to lack of permission in some environments.
	Keywords: `mlx_fwsfx_gen`, /dev/null
	Discovered in Version: 4.33.0-169
	Fixed in Release: 4.34.0-145

MFT Known Issues

The following table provides a list of known issues and limitations.

Internal Ref. No.	Issue
4682265	**Description:** When running FWtrace on Python 3.6 with remote devic
	Workaround: Upgrade Python version to 3.8 and above.
	Keywords: FWtrace, Python
	Discovered in Version: 4.34.0-145
4414021	**Description:** In PCIe Switch-only product, if all PCIe Links are down, th
	Workaround: Remove the device from the power-save mode in order
	Keywords: PCIe
	Discovered in Version: 4.34.0-145
4524862	**Description:** Running mlxfwreset over BlueField-2 and BlueField-3 dev
	Workaround: Use the needed flags for the sync and not rely on the de
	Keywords: mlxfwreset, BlueField-2, BlueField-3
	Discovered in Version: 4.33.0-3002
4296168	**Description:** The default reset level is not executed in hot plug system
	Workaround: Using explicitly reset method 1 parameter in CLI. Execut
	Keywords: BlueField-3, DPU Mode
	Discovered in Version: 4.32.0-120
-	**Description:** mlxptrace does not run on SPECTRUM5 due to missing f

Internal Ref. No.	Issue
	Workaround: Use mlxtrace instead of mlxptrace.
	Keywords: mlxptrace, mlxtrace, tracer tools, SPECTRUM5
	Discovered in Version: 4.32.0-120
4294588	**Description:** In some cases, BlueField-3 users might experience a mis The error message is: "-E- The PCI link is still up even after the expect
	Workaround: Ignore the misleading error message.
	Keywords: mlxfwreset, hot reset
	Discovered in Version: 4.31.0-149
4258362	**Description:** When testing the tools using RDMA as the device, they (
	Workaround: RDMA rename feature is supported only after running t
	Keywords: tools, mst, RDMA
	Discovered in Version: 4.31.0-149
4129306	**Description:** Autocomplete feature is not working as expected for Ub functionalities are all working as expected, only autocomplete on Ubu
	Workaround: Execute the following cmd: `ln -snf /usr/etc/bash`
	Keywords: Autocomplete, Ubuntu
	Discovered in Version: 4.30.0-139
4089179	**Description:** When using mlxlink with --show_module(-m) flag, Rx/TX

	Workaround: To get the Power Current values, use mlxreg with PDDR
	page_select=3, module_info_ext=1, local_port=<local_port> and read
	For example:
	`mlxreg -d /dev/mst/hgx-isr1-020:23108,@dev@mst@mt41692_`
	`"port_type=0,plane_ind=0,lp_msb=0,pnat=0,local_port=1,page_select=3,group_opcc`
	`--op "module_info_ext=1"`

	Keywords: mlxlink, Rx/TX Power Current
	Discovered in Version: 4.29.0-131
3262855	**Description:** The mlxfwreset tool might fail when using PPC64LE on t
	Workaround: N/A

Internal Ref. No.	Issue
	Keywords: mlxfwreset, PPC64LE
	Discovered in Version: 4.29.0-131
3926386	**Description:** Secure boot devices, such as ConnectX7 and above, doe
	Workaround: N/A
	Keywords: ConnectX7, PXE/UEFI ROM data, IMG file
	Discovered in Version: 4.29.0-131
3886315	**Description:** '--sync 0' argument must be specified when resetting or
	Workaround: N/A
	Keywords: mlxfwreset, sync 0, ARM
	Discovered in Version: 4.28.0-92
3872303	**Description:** Activation of MMS4X00-NS transceivers may fail with rc
	Workaround: Reset the Switch/HCA to activate the new firmware on
	Keywords: MMS4X00-NS
	Discovered in Version: 4.28.0-92
3743317	**Description:** Reset flow is not supported when Hotplug is enabled. Th
	Workaround: N/A
	Keywords: Reset flow, Hotplug
	Discovered in Version: 4.26.1
3738146	**Description:** mlxfwreset does not support MRSR-6 when using Quant
	Workaround: N/A
	Keywords: mlxfwreset, MRSR-6
	Discovered in Version: 4.26.1
3641618	**Description:** Running a command triggers the following error messag `/lib/libgcc_s.so.1: version GCC_4.5.0 required by /us`
	Workaround: Run the following command: `export LD_LIBRARY_PATH=/usr/local/lib/gcc12:$LD_LIBRA`
	Keywords: libstd, gcc, mft, libgcc
	Discovered in Version: 4.26.0

Internal Ref. No.	Issue
3549141	**Description:** mlxfwreset usage by the Prometheus PCIe switch is curr
	Workaround: N/A
	Keywords: mlxfwreset, Prometheus PCIe switch
	Discovered in Version: 4.25.0
3262855	**Description:** The mlxfwreset tool might fail when using PPC64LE on t
	Workaround: N/A
	Keywords: mlxfwreset, PPC64LE, RH 8.7
	Discovered in Version: 4.25.0
3090162	**Description:** The PCIe Error Injection feature is not supported due to
	Workaround: N/A
	Keywords: PCI Error Injection
	Discovered in Version: 4.22.0
3446066	**Description:** When using ConnectX-7 and later cards, the link should
	Workaround: N/A
	Keywords: mlxlink
	Discovered in Version: 4.23.0
3352983	**Description:** mlxfwreset does not work on mlnx-os/sonic/cumulus.
	Workaround: N/A
	Keywords: mlxfwreset
	Discovered in Version: 4.23.0
3418112	**Description:** Loading a new firmware may require running `mlxfwres`
	Workaround: N/A
	Keywords: mlxfwreset
	Discovered in Version: 4.24.0
3314750	**Description:** When entering link speed values, you can specify a singl suggesting possible values, only works for the first value in a list of va Additionally, the autocompletion list includes all possible speeds. Som Affected shell commands are: `port speed` `port autonegotiation on speed` `port autoneg`

Internal Ref. No.	Issue
	Any inconvenience caused by these limitations will be addressed in fu
	Workaround: When entering a link speed, you may press the key first type a comma (,) to select the next speed. Repeat the process to forn Once the command is fired, the underlying MFT tool will inform you if of the available options.
	Keywords: mft-shell, link-speed
	Discovered in Version: 4.23.0
3188577	**Description:** Some firmware scratchpad registers have been moved t are included in the MFT package. Otherwise, the mstdumps device will not retrieve the firmware versio
	Workaround: N/A
	Keywords: CSV, mstdump
	Discovered in Version: 4.22.0
2787479	**Description:** mlxcables shows the wrong firmware version for OSFP c
	Workaround: N/A
	Keywords: mlxcables, OSFP, firmware version
	Discovered in Version: 4.18.0
2823492	**Description:** mlxfwreset is not supported on DPU with GPU boards.
	Workaround: N/A
	Keywords: mlxfwreset
	Discovered in Version: 4.18.0
2715716	**Description:** mlxfwreset is not supported on secure-boot host device
	Workaround: N/A
	Keywords: mlxfwreset
	Discovered in Version: 4.18.0
2752916	**Description:** The information of the IB/ETH protocols should not be st
	Workaround: N/A
	Keywords: mlxlink
	Discovered in Version: 4.18.0
2838222	**Description:** mlxfwreset is not supported on kernel 3.10.0-1062.el7.x8

Internal Ref. No.	Issue
	Workaround: N/A
	Keywords: mlxfwreset
	Discovered in Version: 4.18.0
2703663	**Description:** Running flint commands on the hypervisor while a Virtua
	Workaround: N/A
	Keywords: flint, kernel, VM
	Discovered in Version: 4.17.0
2670833	**Description:** Burning firmware using DMA might fail on virtual FreeBS
	Workaround: N/A
	Keywords: Firmware burning, DMA, FreeBS, VM
	Discovered in Version: 4.17.0
2484780	**Description:** Configuring TX/RX_rate to 200GbE in test mode fails.
	Workaround: To work with the new speeds specify the number of lan • 100G_1X/200G_2X/400G_4X/800G_8X for NDR speeds • 50G_1X/100G_2X/200G_4X/400G_4X for HDR speeds
	Keywords: 200GbE, Tx/Rx
	Discovered in Version: 4.17.0
2392334	**Description:** Using the MFT with the --with-pcap option to install stec • Libraries and header files for the libpcap library • Libraries and header files for Python development library • Package Installer for Python (PIP) available
	Workaround: To install the third-party dependencies, perform the foll 1. Install `libpcap-devel` or `libpcap-dev` on Debian-based di 2. Install `python3-devel` or `python3-dev` on Debian-based di 3. Bootstrap the PIP installer in one of the following ways: • On Python 3.4 or newer bootstrap it from the standard library b • Install PIP with Linux Package Manager, for more details see: htt

Internal Ref. No.	Issue
	Keywords: stedump utility
	Discovered in Version: 4.16.0
2376425	**Description:** Direct Device Assignment (DDA, ak.a. pass-through) faci
	Workaround: Burn the firmware in PF and then attach the HCA to the
	Keywords: DDA
	Discovered in Version: 4.16.0
2208845/2099263	**Description:** mlxlink does not support test mode for 50GE-KR4 speec
	Workaround: N/A
	Keywords: mlxlink
	Discovered in Version: 4.16.0
-	**Description:** Port toggling with Inband devices using mlxlink fails and
	Workaround: To avoid this issue, perform one of the following options • Use OpenSM (with or without -o) • Use only active ports
	Keywords: Port toggling, mlxlink, Inband devices
	Discovered in Version: 4.14.0-105
2234589	**Description:** For Multi-Host systems, enabling the PRBS test mode ca
	Workaround: Maintain another interface for enabling the link back.
	Keywords: mlxlink
	Discovered in Version: 4.15.0
2167841	**Description:** `"mlxfwmanager --download"` and `"mlxfwmanager`
	Workaround: N/A
	Keywords: mlxup/mlxfwmanager
	Discovered in Version: 4.14.3
2149437	**Description:** When the SLTP configuration is wrongly set, the “Bad sta
	Workaround: N/A
	Keywords: SLTP configuration

Internal Ref. No.	Issue
	Discovered in Version: 4.14.2
1780276	**Description:** "mst server start" runs at foreground instead of the bacl
	Workaround: Use `'&'` --> `'mst server start &'`
	Keywords: 'mst server start', FreeBSD, VMWare ESXi
	Discovered in Version: 4.14.0-105
2001890	**Description:** The argparse module is installed by default in Python ver installed.
	Workaround: N/A
	Keywords: Python, argparse module
	Discovered in Version: 4.13.3
1923665 / 1939791	**Description:** Force Mode does not work when using mlxlink in Connec
	Workaround: N/A
	Keywords: mlxlink, Force Mode, ConnectX-6 IB
	Discovered in Version: 4.13.3
1802662	**Description:** Due to mst signing process, some executions might be s
	Workaround: N/A
	Keywords: mst
	Discovered in Version: 4.13.0
1431471	**Description:** In ConnectX-5 adapter cards, the time-stamp capability
	Workaround: Use the “-d” flag to set the time-stamp.
	Keywords: flint
	Discovered in Version: 4.11.0
1442454	**Description:** Occasionally, when running mstfwreset over a Multi-Hos
	Workaround: N/A
	Keywords: mstfwreset
	Discovered in Version: 4.11.0
-	**Description:** Running mstfwreset on ConnectX-5 Socket-Direct adapt
	Workaround: Reboot the server

Internal Ref. No.	Issue
	Keywords: mstfwreset, ConnectX-5 Socket-Direct
	Discovered in Version: 4.8.0

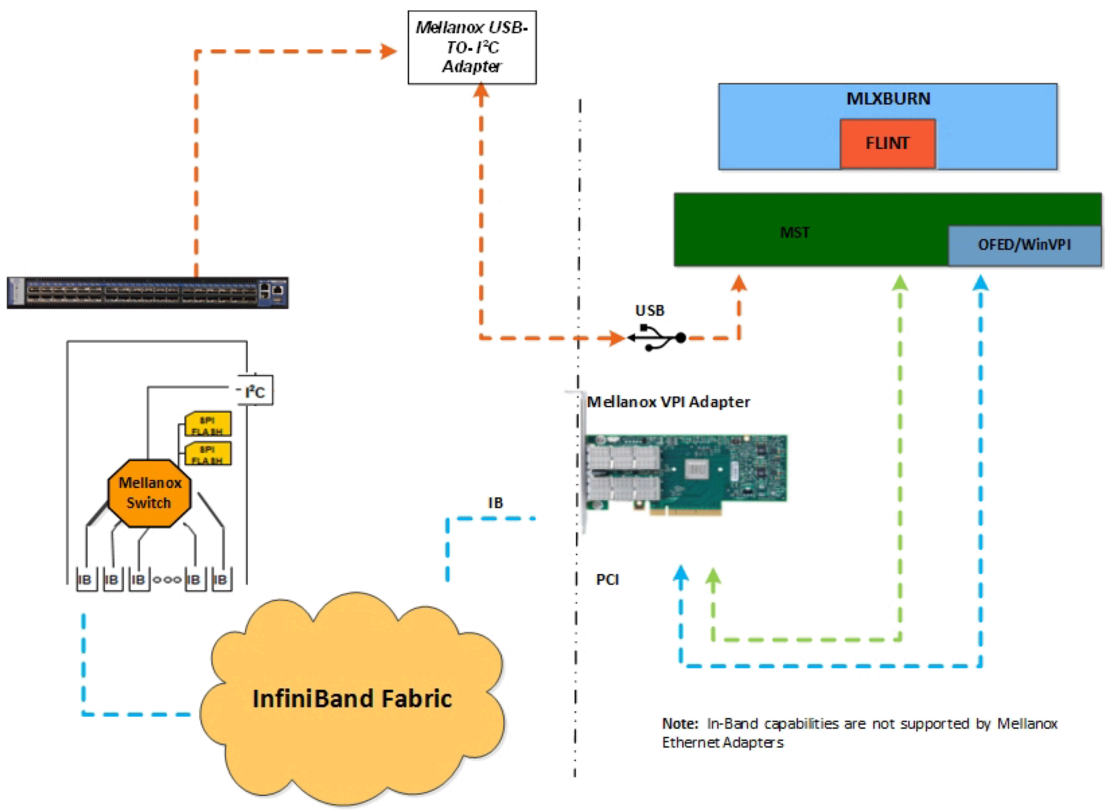
User Manual

MFT package is a set of firmware management and debug tools for NVIDIA® devices. MFT can be used for:

- Generating a standard or customized NVIDIA firmware image
- Querying for firmware information
- Burning a firmware image to a single NVIDIA device

The list of the available tools in the package can be found in the Release Notes document.

MFT – A Scheme of Operation



Supported Operating Systems

Please refer to the release notes of your version for supported operating systems and platforms.

> **Note**
>
> Unless explicitly specified, the usage of the tools is identical for all operating systems.

MFTshell

> **Note**
>
> This is an early alpha release of MFTshell with limited capabilities.

The MFTshell is a frontend for a variety of MFT tools. It is designed to be a starting point for entry-level users to become familiar with the functionality of the tools. In general, commands entered in the shell, result in the execution of the corresponding tool.

You can use the autocompletion feature of MFTshell by pressing the tab key. You may access helpful information by typing ".help" followed by the command on which you wish to receive helpful information.

Access to Hardware Devices

The table below lists the NVIDIA® devices supported by MFT, the supporting tools, and the access methods to these devices.

Device Type	Product Name	HW Access Method			
		PCI	I2C	In-Band	NVIDIA Driver
IB/ETH Network Adapter	NVIDIA ConnectX-8	V	V	V	
	NVIDIA ConnectX-4	V	V	V	
	NVIDIA ConnectX-5	V	V	V	
	NVIDIA ConnectX-5 Ex	V	V	V	

Device Type	Product Name	HW Access Method			
	NVIDIA ConnectX-6	V	V	V	
	NVIDIA ConnectX-6 Dx	V	V	V	
	NVIDIA ConnectX-7	V	V	V	
	NVIDIA BlueField-2	V	V	V	
	NVIDIA BlueField-3	V	V	V	
Ethernet Adapter (NIC)	NVIDIA ConnectX-8	V	V		
	NVIDIA ConnectX-4 Lx	V	V		
	NVIDIA ConnectX-6 Dx	V	V		
	NVIDIA ConnectX-6 Lx	V	V		
	NVIDIA ConnectX-7	V	V		
	NVIDIA BlueField-2	V	V		
Switch	NVIDIA Switch-IB®	V[1]	V	V	
	NVIDIA Switch-IB 2	V[1]	V	V	
	NVIDIA Spectrum™	V	V		
	NVIDIA Spectrum-2	V	V		
	NVIDIA Spectrum-3	V	V		
	NVIDIA Quantum	V	V	V	
GPU	NVIDIA Blackwell				V

Note. V[1] indicates managed switch products only.MFT tools access NVIDIA devices via the PCI Express interface, via a USB to I2C adapter (P/N: MTUSB-1), or via vendor-specific MADs over the InfiniBand fabric (In-Band).

> ⓘ **Note**
>
> In-Band device access requires the local IB port to be in the ACTIVE state and connected to an IB fabric.

All MFT tools address the target hardware device using an *mst device name*. This name is assigned by running the command ‘mst start’ (in Windows, it is not required to run the “mst start” command) for PCI and I2C access. In-Band devices can be assigned by running the 'mst ib add' command. In-Band devices can be assigned by running the 'mst ib add' command.

To list the available mst device names on the local machine, run ‘mst status’. Local PCI devices may also be accessed using device aliases. Supported aliases are:

- PCI device “bus:dev.fn” (e.g. 03:00.0)
- OFED RDMA device (e.g. mlx4_0)
- Network interface with “net-” prefix (e.g. net-eth2)

Run mst status -v to list the devices and their available aliases. The format of an mst device name is as follows:

Via PCI:

```
# mt4099_pci_crX
```

```
# mt4099_pciconf0
```

Where:

X is the index of the adapter on the machine.

_crX devices access the adapter directly (recommended if possible).

_pciconfX devices use configuration cycles to access the adapter.

For example:

```
# mt25418_pci_cr0
```

Via USB to I2C adapter: For example, mtusb-1.

Via Remote device:

```
/dev/mst/mft:23108,@dev@mst@mt4103_pci_cr0
```

Via ibdr device: For example, `/dev/mst/CA_MT4113_server1_HCA-3_ibdr-0,mlx- 5_0,1 or ibdr-0,mlx5_0,1.`

Via In-Band: `<string>lid-<lid_number>.`

For example:

```
/dev/mst/CA_MT4099_mft_HCA-1_lid-0x0002 or simply "lid-2""
```

The “mst ib add” command adds devices in the format:

- For adapters:

  ```
  CA_<device id >_<ib node description>_lid-<lid number>
  ```

- For switches:

  ```
  SW_<device id >_lid-<lid number>
  ```

See Step 3 in .Remote Access to NVIDIA Devices v4.18.1 for instructions on how to obtain the device LID.

Via PCI user level: `<bus:dev.fn>`

For example, if you run `lspci -d 15b3`: NVIDIA devices and PCI Device IDs will be displayed.

```
# /sbin/lspci -d 15b3:
02:00.0 Ethernet controller: Mellanox Technologies Unknown device
6368 (rev a0)
```

VFIO Access:

The purpose of this access is to work in a Kernel lockdown mode, since user space access to the Kernel is not allowed in that environment.

To use it:

1. Load the VFIO driver by running: `modprobe vfio.`
2. Access the device using its DBDF.

Compilation and Installation

Download the relevant MFT package for your OS from the MFT webpage and continue as described in table below according your OS.

OS	Install	Uninstall
Linux	1. Untar the downloaded package 2. Allow packaging of bins to a self-executing file. 3. Run 'install.sh' **For OEM only:** 'install.sh --oem' 4. Start the mst driver by running: mst start NOTE**:** It is possible to customize some installation parameters (such as the target installation path). Run 'install.sh --help' for details.	Uninstall MFT on Linux by running the following command: mft_uninstall.sh

OS	Install	Uninstall
Windows	The installation is EXE based: 1. Double click the EXE file and follow the instructions presented by the installation wizard.	1. Go to Add or remove programs. 2. Remove WinMFT64 depending on the platform type.
FreeBSD	1. Untar the downloaded package. 2. Run “install.sh”	Uninstall MFT on FreeBSD, run the following command: mft_uninstall.sh
VMware	1. Install the package. Run: # esxcli software vib install -v <MST Vib># esxcli software vib install -v <MFT Vib>NOTE**:** For VIBs installation examples, please see below. 2. Reboot system. 3. Start the mst driver. Run: # /opt/mellanox/bin/mst start	1. Uninstall the package. Run: # esxcli software vib remove -n mft 2. Uninstall the mst: • VMKlinux: # esxcli software vib remove -n net-mft• Native: # esxcli software vib remove -n nmst 3. Reboot system.

Example (VIBs installation):

VMK:

```
esxcli software vib install -v /tmp/net-mst_4.6.0.22-
10EM.600.0.0.2295424.vib
esxcli software vib install -v /tmp/mft-4.6.0.22-10EM-
600.0.0.2295424.x86_64.vib
```

Native:

```
esxcli software vib install -v /tmp/nmst-4.6.0.22-
10EM.600.0.0.2295424.x86_64.vib
```

```
esxcli software vib install -v /tmp/mft-4.6.0.22-10EM-
600.0.0.2295424.x86_64.vib
```

Firmware Generation, Configuration, and Update Tools

This chapter contains the following sections:

- mst Service
- MFT Configuration
- mlxfwmanager – Firmware Update and Query Tool
- mlxarchive – Binary Files Compression Tool
- mlxconfig – Changing Device Configuration Tool
- flint – Firmware Burning Tool
- mlxburn – Firmware Image Generator and Burner
- mlxfwreset – Loading Firmware on 5th Generation Devices Tool
- mlxphyburn – Burning Tool for Externally Managed PHY
- mlx_fpga – Burning and Debugging Tool for NVIDIA Devices with FPGA
- cpldupdate – Tool for Programming On-Board CPLDs on NVIDIA Devices
- mlxprivhost - NIC Configuration by the Host Restriction Tool (Zero Trust Mode)
- mlxtokengenerator - Token Creation Tool
- mlxdpa – DPA Applications Sign Tool
- mlxcableimgen - Cable Firmware Image Wrapper Generation Tool
- nvredfish 1

- nvredfish

mst Service

This section contains:

- Linux
- Running mst in an Environment without a Kernel
- Windows
- FreeBSD
- VMware ESXi

Linux

This script is used to start mst service and to stop it. It is also used in other operations with NVIDIA devices, such as in resetting or enabling remote access.

mst Synopsis - Linux

```
mst <command> [switches]
```

mst Commands and Switches Description - Linux

mst start [--with_msix] [--with_unknown] [--with_i2cdev] [--with_lpcdev] [--with_fpga] [--with_fpga_fw_access	Create special files that represent NVIDIA devices in directory /dev/mst. Load appropriate kernel modules and saves PCI configuration headers in directory /var/mst_pci. After successfully completion of this command the MST driver is ready to work. You can configure the start command by editing the configuration file: /etc/mft/mst.conf, for example you can rename you devices. Options: • --with_msix: Create the msix device.

	• --with_unknown: Do not check if the device ID is supported. • --with_i2cdev: Create Embedded I2C master • --with_fpga: Create MST device for the attached FPGA card (Access via Driver) • --with_fpga_fw_access: Create an extended MST device for the attached FPGA (Access via Firmware)
mst stop [--force]	Stop the MST driver service, remove all special files/ directories and unload kernel modules. Options: • --force: Force try to stop mst driver even if it's in use.
mst restart [--with_msix] [--with_unknown] [--with_i2cdev] [--with_lpcdev] [--with_fpga] [--with_fpga_fw_access]	Just like "mst stop" followed by "mst start [--with_msix] [--with_unknown] [--with_i2cdev] [--with_lpcdev] [-- with_fpga] [--with_fpga_fw_access]"
mst server start [port-p <port> [-s <passphrase>]	Start mst MST server to allow incoming connection. The default port is 23108. Use the '-s' flag to define the passphrase used by the server.If no passphrase is provided, a random one will be generated.
mst server stop	Stop the mst server.
mst remote add <hostname>[:port] [-s <passphrase>]	Establish a connection with a specified host on a specified port. The default port is 23108. Add devices on a remote peer to a local devices list.<hostname> may be the host name as well as an IP address.Use the '-s' flag to provide the host's passphrase.If no passphrase is provided, you will be prompted to insert one.
mst remote del <hostname>[:port]	Remove all remote devices on specified hostname. <hostname>[:port] should be specified exactly as in the "mst remote add" command.
mst ib add [OPTIONS] [local_hca_id] [local_hca_port]	Add devices found in the IB fabric for inband access. Requires OFED installation and an active IB link. If local_hca_id and local_hca_port are given, the IB subnet connected to the given port is scanned. Otherwise, the default subnet is scanned. Options:

	• --discover-tool <discover-tool>: The tool that is used to discover the fabric Supported tools: ibnetdiscover, ibdiagnet. default: ibdiagnet • --add-non-mlnx: Add non NVIDIA nodes • --topo-file <topology-file>: A prepared topology file which describes the fabric. For ibnetdiscover: provide an output of the tool. For ibdiagnet: provide LST file that ibdiagnet generates • --use-ibdr: Access by direct route MADs. Available only when using ibnetdiscover tool, for SwitchX and ConnectIB devices NOTE: If a topology file is specified, device are taken from it. Otherwise, a discover tool is run to discover the fabric. • --planarized : Add planarized device NOTE: this device is only supported by mlxlink tool.
mst ib del	Remove all inband devices.
mst cable add [OPTIONS] [params]	Add the cables that are connected to 5th generation devices. There is an option to add the cables found in the IB fabric for Cable Info access, requires WinOF-2 installation and active IB links. If local_hca_id and local_hca_port are given, the IB subnet connected to the given port is scanned. Otherwise, all the devices will be scanned. Options: --with_ib: Add the inband cables in addition to the local PCI devices. params: [local_hca_id] [local_hca_port]
mst cable del	Remove all cable devices.
mst i2c add [--with_retimer] [--with_chipset]	Add I2C devices for each i2c secondary address.
mst gpu add	Add GPU device via NVIDIA Driver.
mst gpu del	Remove all GPU devices.

mst status	Print current status of NVIDIA devices Options: • -v run with high verbosity level (print more info on each device)
mst save	Save PCI configuration headers in directory /var/mst_pci.
mst load	Load PCI configuration headers from directory /var/mst_pci.
mst version	Print the version info

Using mst.conf File in Linux

Edit the /etc/mft/mst.conf configuration file to configure the start operation in Linux (only).

The configuration file consists of lines of rules. Every line will be a rule for mst start. It must be valid, and the rules should be unique. There should also be no duplication of new names and/or serials.

The rule general format is the following:

`$OPCODE $PARAMS`

mst start Supported OPCODES

OPCODE	Definition	Description
RENAME	renames mst devices	• Rule format: RENAME $TYPE $NEW_NAME $ID • Supported types: # MTUSB (where $ID is the iSerial) • Example: RENAME USB my 0x2A4C. • Effect: MTUSB with serial 0x2A4C will be renamed to /dev/mst/mymtusb-1 (always mtusb-1 will be concatenated to the new name).

Examples of mst Usage - Linux

To start the mst driver service:

```
# mst start
```

```
Starting MST (Mellanox Software Tools) driver set
Loading MST PCI module - Success
Loading MST PCI configuration module - Success
Create devices
MTUSB-1 USB to I2C Bridge - Success
```

To stop the service:

```
Success# mst stop
Stopping MST (Mellanox Software Tools) driver set
Unloading MST PCI module - Success
```

To print the current status of NVIDIA devices:

```
# mst status
MST modules:
------------
   MST PCI module loaded
   MST PCI configuration module loaded

MST devices:
------------
/dev/mst/mt4099_pciconf0             - PCI configuration cycles
access.

domain:bus:dev.fn=0000:0b:00.0 addr.reg=88 data.reg=92
                                       Chip revision is: 01
/dev/mst/mt4099_pci_cr0              - PCI direct access.

domain:bus:dev.fn=0000:0b:00.0 bar=0xd2600000 size=0x100000
                                       Chip revision is: 01
/dev/mst/mtusb-1                     - USB to I2C adapter as I2C
master
```

```
                                         iSerial = 0x1683
```

To show the devices status with detailed information

```
# mst status -v
PCI devices:
DEVICE_TYPE         MST                           PCI         RDMA
NET         NUMA
ConnectX4(rev:0)    /dev/mst/mt4115_pciconf0      08:00.0     mlx5_0
net-ib2     -1
ConnectX4(rev:0)    /dev/mst/mt4115_pciconf0.1    08:00.1     mlx5_1
net-ib3     -1
ConnectIB(rev:0)    /dev/mst/mt4113_pciconf0      0b:00.0     mlx5_2
net-ib4,

net-ib5     -1
ConnectX3(rev:1)    /dev/mst/mt4099_pciconf0
ConnectX3(rev:1)    /dev/mst/mt4099_pci_cr0       0e:00.0     mlx4_0
net-ib0,

net-ib1     -1
I2C devices:
-------------------
MST                 Serial
/dev/mst/mtusb-1    0x1B5C
```

In case the device has Function Per Port (FPP) enabled on it, a new device will appear in the `mst status -v` output with information about the second physical function. Example:

```
DEVICE_TYPE         MST                           PCI         RDMA
NET         NUMA
ConnectX4(rev:0)    /dev/mst/mt4115_pciconf0      07:00.0     mlx5_4
net-ib4     -1
```

```
ConnectX4(rev:0)    /dev/mst/mt4115_pciconf0.1     07:00.1     mlx5_5
net-ib5     -1
```

Running mst in an Environment without a Kernel

mst can work even without kernel module being installed on the machine or if the kernel is down. In this case, the devices' names will be the PCI address of the devices.

Example:

```
> mst status
MST modules:
------------
   MST PCI module is not loaded
   MST PCI configuration module is not loaded

PCI Devices:
------------
05:00.0
08:00.0
82:00.0

> mst status -v
MST modules:
------------
   MST PCI module is not loaded
   MST PCI configuration module is not loaded


PCI devices:
------------
```

```
DEVICE_TYPE              MST   PCI          RDMA     NET
NUMA
ConnectX3Pro(rev:0)  NA    05:00.0    mlx4_0   net-ib0,net-ib1
ConnectX4(rev:0)     NA    08:00.0    mlx5_0   net-ib2
ConnectX4(rev:0)     NA    08:00.1    mlx5_1   net-ib3
ConnectIB(rev:0)     NA    82:00.0    mlx5_2   net-ib4,net-ib5
```

> **Note**
>
> The MST interface will be NA in mst status -v[v] output.

Run commands with these devices:

```
> flint -d 08:00.0 q
Image type:          FS3
FW Version:          12.16.0048
FW Release Date:     14.3.2016
Description:         UID                 GuidsNumber
Base GUID:           7cfe90030029205e         4
Base MAC:            00007cfe9029205e         4
Image VSD:
Device VSD:
PSID:                MT_2190110032
```

Windows

mst Synopsis - Windows

mst status [-v] | help | server <start|stop> | ib <add|del> | version | remote <add|del> <hostname>

mst Commands and Switches Description - Windows

Note

There are no mst start or stop operations in Windows.

mst server start [port]	Start mst server to allow incoming connection. Default port is 23108.
mst server stop	Stop mst server.
mst remote add <hostname>:[port] [-s <passphrase>]	Establish a connection with a specified host on a specified port (the default port is 23108). Add devices on a remote peer to a local devices list. <hostname> may be host name as well as an IP address Use the '-s' flag to provide the host's passphrase. If no passphrase is provided, you will be prompted to insert one. **Note:** [-s] is applicable when the host server OS is not windows.
mst remote del <hostname>[:port]	Remove all remote devices on a specified hostname <hostname>:<port> should be specified exactly as in the "mst remote add" command.
mst ib del	Remove all inband devices.
mst cable add [OPTIONS] [params]	Add the cables that are connected to 5th generation devices. There are an option to add the cables found in the IB fabric for Cable Info access, requires OFED installation and active IB links.If local_hca_id and local_hca_port are given, the IB subnet connected to the given port is scanned. Otherwise, all the devices will be scanned.OPTIONS:--with_ib: Add the inband cables in addition to the local PCI devices.params: [local_hca_id] [local_hca_port]
mst cable del	Remove all cable devices.
mst status	Print current status of NVIDIA devices.
mst version	Print the version info.
mst ib add [OPTIONS] [local_hca_id]	Add devices found in the IB fabric for inband access.

[local_hca_port] [lst-file]	Requires MLNX_WinOF installation and an active IB link.If local_hca_id and local_hca_port are given, the IB subnet connected to the given port is scanned. Otherwise, the default subnet is scanned.If an lst-file is specified, devices are taken from this file. Otherwise, ibnetdiscover tool is run to discover the fabric. Options: • --discover-tool <discover-tool>: The tool used to discover the fabric. Supported tools: ibnetdiscover, ibdiagnet. default: ibnetdiscover NOTE: The discover tool argument is intended only for parsing purposes, thus you need to specify an lst-file with it. • --add-non-mlnx: Add non NVIDIA nodes. • --use-ibdr: Access by direct route MADs. Available only when using ibnetdiscover tool, for SwitchX and ConnectIB devices. • --no-format-check: Do not check the format of the given local_hca_id. The expected format of the local_hca_id is: ibv_device[0-9]+. • --topo-file <topology-file>: A prepared topology file which describes the fabric. For ibnetdiscover: provide an output of the tool. For ibdiagnet: provide and lst-file that ibdiagnet generates. NOTE: If a topology file is specified, the devices are taken from it. Otherwise, a discover tool is run to discover the fabric.
mst help	Print this help information.

Examples of mst Usage - Windows

To print the current status of NVIDIA devices:

```
# mst status
MST devices:
------------
mt4115_pciconf0
```

```
mtusb-1
mtusb-2
```

To show the devices status with detailed information:

```
# mst status -v

MST devices:
------------
mt4115_pciconf0   bus:dev.fn=13:00.0
mt4115_pciconf0.1 bus:dev.fn=13:00.1
mtusb-1           iSerial=0x1ccc
mtusb-2           iSerial=0x1cd5
```

FreeBSD

mst Synopsis - FreeBSD

```
mst <command> [switches]
```

Commands and Switches Description - FreeBSD

Note

There are no mst start or stop operations in FreeBSD.

mst status	Print current status of NVIDIA devices.
mst help	Print this help information.
mst version	Print mst version information.
mst server start [port]	Start mst server to allow incoming connection. Default port is 23108.
mst server stop	Stop mst server.
mst cable add	Add the cables that are connected to the device
mst cable del	Delete the added cables

Examples of mst Usage - FreeBSD

To print the current status of NVIDIA devices:

```
VMwareMST devices:
------------
pci0:3:0:0 - MT27500 Family [ConnectX-3]
```

> **Note**
>
> The mst status output is taken from parsing the `pciconf` output.

To show the devices status with detailed information:

```
# mst status -v

PCI devices:
------------
DEVICE_TYPE             MST   PCI            RDMA       NET        NUMA
```

```
ConnectX4LX(rev:0)          pci0:2:0:0  02:00.0   mlx5_0
ConnectX4LX(rev:0)          pci0:2:0:1  02:00.1   mlx5_1
ConnectX4(rev:0)            pci0:3:0:0  03:00.0   mlx5_2
```

VMware ESXi

mst Synopsis - VMware

```
mst <command> [switches]
```

Commands and Switches Description - VMwareMST

mst start	Create special files that represent NVIDIA devices in directory/dev. Load appropriate modules. After successfully completing this command, the mst driver will be ready to work.
mst stop	Stop NVIDIA mst driver service and unload the kernel modules.
mst restart	"mst stop" followed by "mst start"
mst server start [-p|--port portport] [-s <passphrase>]	Start the MST server to allow incoming connection. The default port is 23108. Use the '-s' flag to define the passphrase used by the server.If no passphrase is provided, a random one will be generated.
mst server stop	Stop the mst server.
mst status	Print current status of NVIDIA devices Options:-v run with a high verbosity level (print more info on each device)
mst version	Print the version info

Examples of mst Usage - VMware

To print the current status of NVIDIA devices:

Native

```
# /opt/mellanox/bin/mst status
MST devices:
------------
mt4099_pciconf0
mt4099_pci_cr0
```

VMK Linux

```
# /opt/mellanox/bin/mst status
MST devices:
/dev/mt4099_pciconf0
/dev/mt4099_pci_cr0
```

To show the devices status with detailed information:

```
# /opt/mellanox/bin/mst status -vv
PCI devices:
DEVICE_TYPE         MST                  PCI         RDMA     NET
NUMA
ConnectX4(rev:0)    mt4115_pciconf0     03:00.0              net-
vmnic4
ConnectX4(rev:0)    mt4115_pciconf0.1   03:00.1              net-
vmnic5
ConnectX3Pro(rev:0) mt4103_pci_cr0      05:00.0              net-
vmnic1,netvmnic1000102
ConnectX3Pro(rev:0) mt4103_pciconf0     05:00.0              net-
vmnic1,netvmnic1000102
```

For further information on In-Band and Remote Access, please refer to In-Band Access to Multiple IB Subnets, Accessing Remote InfiniBand Device by Direct Route MADs, and Remote Access to Device by Sockets.

MFT Configuration

MFT configuration file resides in `/etc/mft/mft.conf`. It includes a list of defines and their values in the following syntax: <DEF> = <value>.

> **Note**
>
> This capability is available in Linux only.

MKey Configuration

In order to use the mft tools when the MKEY is configured, please edit the /etc/mft.conf file as shown below:

- mkey_enable=yes (default: no)
- sm_config_dir= (if empty, the SM config directory will be: `/var/cache/opensm/`)
- sm_conf_file_path=<opensm configuration file full path> (default `/etc/opensm/opensm.conf`)

mlxfwmanager – Firmware Update and Query Tool

The mlxfwmanager is a firmware update and query utility which scans the system for available NVIDIA devices (only mst PCI devices) and performs the necessary firmware updates.

For further information on firmware update, please refer to Booting HCA Device in Livefish Mode.

Note

The examples throughout the document use pci “bus.dev.fn” format. However, all the examples are inter-changeable with the mlxfwmanager -d /dev/mst/<device> format.

mlxfwmanager Synopsis

```
# [-d|--dev DeviceName] [-h|--help] [-v|--version] [--query] [--query-format Format] [-u|--update] [-i|--image-file FileName] [-D|--image-dir DirectoryName] [-f|--force] [-y|--yes] [--no] [--clear-semaphore] [--exe-rel-path] [-l|--list-content] [--archive-names] [--nofs] [--log] [-L|--log-file LogFileName] [--no-progress] [-o|--outfile OutputFileName] [--online] [--online-query-psid PSIDs] [--key key] [--download DirectoryName] [--download-default] [--get-download-opt OPT] [--download-device Device] [--download-os OS] [--download-type Type] [--ssl-certificate Certificate] [--no_fw_ctrl] [--component_type componentName] [--skip_if_same]
```

where:

-d|--dev DeviceName	Perform operation for specified mst device(s). Run 'mst status' command to list the available devices. Multiple devices can be specified delimited by semicolons. A device list containing semicolons must be quoted.
-h|--help	Show this message and exit.
-v|--version	Show the executable version and exit.
--query	Query device(s) info

--query-format Format	(Query|Onlinequery)outputformat,XML|Text-defaultText
-u|--update	Update firmware image(s) on the device(s).
-i|--image-file FileName	Specified image file to use.
-D|--image-dir DirectoryName	Specified directory instead of default to locate image files.
-f|--force	Force image update
-y|--yes	Answer is yes in prompts
--no	Answer is no in prompts
--clear-semaphore	Force clear the flash semaphore on the device, No command is allowed when this flag is used. NOTE: May result in system instability or flash corruption if the device or another application is currently using the flash. Exercise caution.
--exe-rel-path	Use paths relative to the location of the executable
-l|--list-content	List file/Directory content, used with --image-dir and --image-file flags
--archive-names	Display archive names in listing
--nofs	Burn image in a non failsafe manner
--log	Create log file
-L|--log-file LogFileName	Use specified log file
--no_fw_ctrl	Do not use firmware Ctrl update
--no-progress	Do not show progress
-o|--outfile OutputFileName	Write to specified output file
--online	Fetch required FW images online from NVIDIA server
--online-query-psid PSIDs	Query FW info, PSID(s) are comma separated
--key key	Key for custom download/update
--download DirectoryName	Download files from server to a specified directory
--download-default	Use Default values for download
--get-download-opt OPT	Get download options for OS or Device Options are: OS, Device

--download-device Device	Use '--get-download-opt Device' option to view available devices for device specific downloads
--download-os OS	Only for self_extractor download: Use '--get-download-opt OS' option to view available OS for sfx download
--download-type Type	MFA | self_extractor - default All
--ssl-certificate Certificate	SSL certificate for secure connection
--component_type componentName	Specify the PLDM component type to extract
--skip_if_same	Skip firmware update if current and new versions match

Querying the Device

To query a specific device, use the following command line:

```
# mlxfwmanager -d <device> --query
```

To query all the devices on the machine, use the following command line:

```
# mlxfwmanager --query
```

Examples:

Query the device.

```
mlxfwmanager -d 18:00.0 --query
Querying Mellanox devices firmware ...



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

```
  Device Type:      ConnectX7
  Part Number:      CX713106AE-HEA_QP1_Ax
  Description:      NVIDIA ConnectX-7 HHHL adapter Card; 200GbE
(default mode) / NDR200 IB; Dual-port QSFP112; PCIe 5.0 x16 with x16
PCIe extension option; Crypto Enabled; Secure Boot Capable; IPN
for internal use
  PSID:             MT_0000000894
  PCI Device Name:  18:00.0
  Base MAC:         1070fd875230
  Versions:         Current        Available
     FW             28.43.0180     28.45.1008
     PXE            N/A            3.7.0500
     UEFI           N/A            14.38.0016



  Status:           Update required
---------
Found 1 device(s) requiring firmware update. Please use -u flag
to perform the update.
```

Query all the devices.

```
Querying Mellanox devices firmware ...



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

```
  Device Type:        ConnectX6
  Part Number:        MCX653106A-ECA_Ax
  Description:        ConnectX-6 VPI adapter card; H100Gb/s
(HDR100; EDR IB and 100GbE); dual-port QSFP56; PCIe3.0 x16; tall
bracket; ROHS R6
  PSID:               MT_0000000224
  PCI Device Name:    0000:04:00.0
  Base GUID:          b8599f030023f954
  Versions:           Current        Available
     FW               20.43.1014     20.43.1014
     PXE              3.7.0500       3.7.0500
     UEFI             14.36.0016     14.36.0016



  Status:             Up to date



Device #2:
----------



  Device Type:        ConnectX6LX
  Part Number:        MCX631102AN-ADA_Ax
  Description:        ConnectX-6 Lx EN adapter card; 25GbE ; Dual-
port SFP28; PCIe 4.0 x8; No Crypto
  PSID:               MT_0000000531
  PCI Device Name:    0000:07:00.0
  Base GUID:          0c42a1030060614c
  Base MAC:           0c42a160614c
  Versions:           Current        Available
     FW               26.45.0386     26.45.1008
     PXE              3.7.0500       3.7.0500
```

```
    UEFI              14.38.0014      14.38.0016



  Status:             Update required



Device #3:
----------



  Device Type:        ConnectX5
  Part Number:        MCX556A-ECA_Ax
  Description:        ConnectX-5 VPI adapter card; EDR IB (100Gb/s)
and 100GbE; dual-port QSFP28; PCIe3.0 x16; tall bracket; ROHS R6
  PSID:               MT_0000000008
  PCI Device Name:    0000:21:00.0
  Base GUID:          248a0703008db026
  Versions:           Current        Available
    FW                16.35.3004     16.35.3004
    PXE               N/A            3.6.0902
    UEFI              N/A            14.29.0015



  Status:             Up to date



Device #4:
----------


```

```
  Device Type:      ConnectX4
  Part Number:      MCX455A-FCA_Ax
  Description:      ConnectX-4 VPI adapter card; FDR IB (56Gb/s)
and 40GbE; single-port QSFP28; PCIe3.0 x16; ROHS R6
  PSID:             MT_2160111021
  PCI Device Name:  0000:24:00.0
  Base GUID:        8e0b01e681d98270
  Versions:         Current        Available
     FW             12.28.2006     12.28.2006
     PXE            3.6.0102       3.6.0102
     UEFI           14.21.0017     14.21.0017



  Status:           Up to date



Device #5:
----------




  Device Type:      ConnectX6LX
  Part Number:      MCX631102AN-ADA_Ax
  Description:      ConnectX-6 Lx EN adapter card; 25GbE ; Dual-
port SFP28; PCIe 4.0 x8; No Crypto
  PSID:             MT_0000000531
  PCI Device Name:  0000:0a:00.0
  Base GUID:        0c42a10300a981fe
  Base MAC:         0c42a1a981fe
  Versions:         Current        Available
     FW             26.45.0386     26.45.1008
     PXE            3.7.0500       3.7.0500
     UEFI           14.38.0014     14.38.0016
```

```
  Status:           Update required

---------
Found 2 device(s) requiring firmware update. Please use -u flag
to perform the update.
```

Query XML:


```
mlxfwmanager --query-format xml
<Devices>
        <Device pciName="0000:04:00.0" type="ConnectX6" psid="MT_0000000224"
partNumber="MCX653106A-ECA_Ax">
                <Versions>
                        <FW current="20.43.1014" available="20.43.1014"/>
                        <PXE current="3.7.0500" available="3.7.0500"/>
                        <UEFI current="14.36.0016"
available="14.36.0016"/>
                </Versions>
                <GUIDs Base_Guid="b8599f030023f954" />
                <Status>Up to date</Status>
                <Description>ConnectX-6 VPI adapter card;
H100Gb/s (HDR100; EDR IB and 100GbE); dual-port QSFP56; PCIe3.0
x16; tall bracket; ROHS R6</Description>
        </Device>
        <Device pciName="0000:07:00.0" type="ConnectX6LX" psid="MT_0000000531"
partNumber="MCX631102AN-ADA_Ax">
                <Versions>
                        <FW current="26.45.0386" available="26.45.1008"/>
                        <PXE current="3.7.0500" available="3.7.0500"/>
```

```
                        <UEFI current="14.38.0014"
available="14.38.0016"/>
                </Versions>
                <MACs Base_Mac="0c42a160614c" />
                <GUIDs Base_Guid="0c42a1030060614c" />
                <Status>Update required</Status>
                <Description>ConnectX-6 Lx EN adapter card; 25GbE
; Dual-port SFP28; PCIe 4.0 x8; No Crypto</Description>
        </Device>
        <Device pciName="0000:21:00.0" type="ConnectX5" psid="MT_0000000008"
partNumber="MCX556A-ECA_Ax">
                <Versions>
                        <FW current="16.35.3004" available="16.35.3004"/>
                        <PXE current="N/A" available="3.6.0902"/>
                        <UEFI current="N/A" available="14.29.0015"/>
                </Versions>
                <GUIDs Base_Guid="248a0703008db026" />
                <Status>Up to date</Status>
                <Description>ConnectX-5 VPI adapter card; EDR IB
(100Gb/s) and 100GbE; dual-port QSFP28; PCIe3.0 x16; tall
bracket; ROHS R6</Description>
        </Device>
        <Device pciName="0000:21:00.0" type="ConnectX5" psid="MT_0000000008"
partNumber="MCX556A-ECA_Ax">
                <Versions>
            <FW current="16.35.3004" available="16.35.3004"/>
            <PXE current="N/A" available="3.6.0902"/>
            <UEFI current="N/A" available="14.29.0015"/>
        </Versions>
        <GUIDs Base_Guid="248a0703008db026" />
        <Status>Up to date</Status>
        <Description>ConnectX-5 VPI adapter card; EDR IB
(100Gb/s) and 100GbE; dual-port QSFP28; PCIe3.0 x16; tall
bracket; ROHS R6</Description>
    </Device>
</Devices>
```

Archived Images Content

Supports listing the contents of images archive.

- When running this command, the tool will list all firmware images within this PLDM package for each image it displays.

  Usage:

```
mlxfwmanager -i <pldm-path> --list-content
```

- When running this command, the tool will list all firmware images within this mfa package.

  Usage:

  ```
  mlxfwmanager -i <mfa-file> --list-content
  ```

For each image, it displays the following: PSID, Part Number, firmware version, and device description.

Updating the Device

To update a device on the machine, use the following command line: (Note: If only PXE rom needs update, please add -f to the command line.)

```
# mlxfwmanager -u -d <device> -i <existingMFAFile>
```

Update the device using PLDM:

```
mlxfwmanager -d <device> -i <pldm_package>
```

Example:

```
mlxfwmanager -u -d 18:00.0 -i /tmp/fw-ConnectX7-rel-28.45.1008.mfa
Querying Mellanox devices firmware ...

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


  Device Type:      ConnectX7
  Part Number:      CX713106AE-HEA_QP1_Ax
  Description:      NVIDIA ConnectX-7 HHHL adapter Card; 200GbE
(default mode) / NDR200 IB; Dual-port QSFP112; PCIe 5.0 x16 with x16
PCIe extension option; Crypto Enabled; Secure Boot Capable; IPN
for internal use
  PSID:             MT_0000000894
  PCI Device Name:  18:00.0
  Base MAC:         1070fd875230
  Versions:         Current        Available
     FW             28.43.0180     28.45.1008
     PXE            N/A            3.7.0500
     UEFI           N/A            14.38.0016

  Status:           Update required

---------
Found 1 device(s) requiring firmware update…
```

Updating the Device Online

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

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

Example:

```
mlxfwmanager --online -u -d 21:00.0 -y
Querying Mellanox devices firmware ...

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

  Device Type:      ConnectX7
  Part Number:      MCX75210AAS-NEA_Ax
  Description:      NVIDIA ConnectX-7 HHHL Adapter Card; NDR IB;
Single-port OSFP; PCIe 5.0 2x8 in a row (Socket Direct); Crypto
Disabled; Secure Boot Enabled
  PSID:             MT_0000000851
  PCI Device Name:  /dev/mst/mt4129_pciconf0
  Base GUID:        N/A
  Base MAC:         N/A
  Versions:         Current        Available
     FW             28.43.1014     28.43.1014
     PXE            3.7.0500       N/A
     UEFI           14.36.0016     N/A

  Status:           Forced update required

---------
Found 1 device(s) requiring firmware update...


Please wait while downloading MFA(s) 100%
Device #1: Updating FW ...
FSMST_INITIALIZE -   OK
```

```
Writing Boot image component -   OK
Done

Restart needed for updates to take effect.
```

Downloading Firmware Images and Firmware Update Packages

To download firmware images/firmware update packages, use the following command line:

```
mlxfwmanager --download <DownloadDir> --download-device
<DeviceType> --download-os <OS> --download-type <DownloadType>
```

To get the list of the supported devices or OSes, use the flag "--get-download-opt OPT"

```
mlxfwmanager --get-download-opt OS
esxi_6_5_native
esxi_6_native
fbsd10_64
linux
linux_arm64
linux_ppc64
linux_ppc64le
linux_x64
windows
windows_x64

mlxfwmanager --get-download-opt Device
All
```

Examples:

Downloading Firmware Images/Firmware Update Packages:

```
mlxfwmanager --download /tmp/DownloadDir --download-device All --
download-os All --download-type self_extractor

------ Files To Be Downloaded ------

All :
-----

<Files>:
0 - linux_x64/mlxup
1 - windows/mlxup.exe
2 - esxi_6_native/mlxup
3 - windows_x64/mlxup.exe
4 - linux_ppc64/mlxup
5 - linux_arm64/mlxup
6 - linux_ppc64le/mlxup
7 - linux/mlxup
8 - fbsd10_64/mlxup
9 - esxi_6_5_native/mlxup
<Release Notes>:
For more details, please refer to the following FW release notes:
1- ConnectX3 (2.42.5000): http://www.mellanox.com/pdf/firmware/ConnectX3-FW-
2_42_5000-release_notes.pdf
2- ConnectX3Pro (2.42.5000): http://www.mellanox.com/pdf/firmware/ConnectX3-Pro-FW-
2_42_5000-release_notes.pdf
3- Connect-IB (10.16.1200): http://www.mellanox.com/pdf/firmware/ConnectIBFW-
10_16_1200-release_notes.pdf
4- ConnectX4 (12.28.2008):
https://docs.mellanox.com/display/ConnectX4Firmwarev12282006
5- ConnectX4Lx (14.30.1004):
https://docs.mellanox.com/display/ConnectX6LxFirmwarev14301004
6- ConnectX5 (16.30.1004):
https://docs.mellanox.com/display/ConnectX5Firmwarev16301004
```

```
7- ConnectX6 (20.30.1004):
https://docs.mellanox.com/display/ConnectX6Firmwarev20301004
8- ConnectX6Dx (22.30.1004):
https://docs.mellanox.com/display/ConnectX6DxFirmwarev22301004
9- ConnectX6Lx (26.30.1004):
https://docs.mellanox.com/display/ConnectX6LxFirmwarev26301004
10- BlueField (18.30.1004): N/A
11- BlueField2 (24.30.1004): N/A


Perform Download? [y/N]: y
Please wait while downloading Files to : '/tmp/DownloadDir'
0 - linux_x64/mlxup : Done
1 - windows/mlxup.exe : Done
2 - esxi_6_native/mlxup : Done
3 - windows_x64/mlxup.exe : Done
4 - linux_ppc64/mlxup : Done
5 - linux_arm64/mlxup : Done
6 - linux_ppc64le/mlxup : Done
7 - linux/mlxup : Done
8 - fbsd10_64/mlxup : Done
9 - esxi_6_5_native/mlxup : Done

Downloading file(s) to : '/tmp/DownloadDir' is done successfully
```

Downloading firmware images/firmware update packages using custom key:

```
mlxfwmanager --download /tmp/DownloadDir --download-device All --
download-os All --download-type All --key last_release

------ Files To Be Downloaded ------


All :
```

```
-----

<Files>:
0 - Mellanox_Firmware_20210407.mfa
1 - linux_x64/mlxup
2 - windows/mlxup.exe
3 - esxi_6_native/mlxup
4 - windows_x64/mlxup.exe
5 - linux_ppc64/mlxup
6 - linux_arm64/mlxup
7 - linux_ppc64le/mlxup
8 - linux/mlxup
9 - fbsd10_64/mlxup
10 - esxi_6_5_native/mlxup

<Release Notes>:
For more details, please refer to the following FW release notes:
For more details, please refer to the following FW release notes:
1- ConnectX3 (2.42.5000): http://www.mellanox.com/pdf/firmware/ConnectX3-FW-2_42_5000-release_notes.pdf
2- ConnectX3Pro (2.42.5000): http://www.mellanox.com/pdf/firmware/ConnectX3-Pro-FW-2_42_5000-release_notes.pdf
3- Connect-IB (10.16.1200): http://www.mellanox.com/pdf/firmware/ConnectIBFW-10_16_1200-release_notes.pdf
4- ConnectX4 (12.28.2008):
https://docs.mellanox.com/display/ConnectX4Firmwarev12282006
5- ConnectX4Lx (14.30.1004):
https://docs.mellanox.com/display/ConnectX6LxFirmwarev14301004
6- ConnectX5 (16.30.1004):
https://docs.mellanox.com/display/ConnectX5Firmwarev16301004
7- ConnectX6 (20.30.1004):
https://docs.mellanox.com/display/ConnectX6Firmwarev20301004
8- ConnectX6Dx (22.30.1004):
https://docs.mellanox.com/display/ConnectX6DxFirmwarev22301004
9- ConnectX6Lx (26.30.1004):
https://docs.mellanox.com/display/ConnectX6LxFirmwarev26301004
```

```
10- BlueField (18.30.1004): N/A
11- BlueField2 (24.30.1004): N/A

Perform Download? [y/N]: y
Please wait while downloading Files to : '/tmp/DownloadDir'
0 - Mellanox_Firmware_20210407.mfa : Done
1 - linux_x64/mlxup : Done
2 - windows/mlxup.exe : Done
3 - esxi_6_native/mlxup : Done
4 - windows_x64/mlxup.exe : Done
5 - linux_ppc64/mlxup : Done
6 - linux_arm64/mlxup : Done
7 - linux_ppc64le/mlxup : Done
8 - linux/mlxup : Done
9 - fbsd10_64/mlxup : Done
10 - esxi_6_5_native/mlxup : Done

Downloading file(s) to : '/tmp/DownloadDir' is done successfully
```

UPMF

Update Package for NVIDIA Firmware (UPMF) is a new method used to distribute firmware to end users. Instead of providing multiple binary files (one for each board type) and burning them using the flint tool, the UPMF method requires only a single standalone file.

The UPMF is a self-extracting executable that contains a set of firmware binary images, and the mlxfwmanager firmware update tool.

UPMF provides:

- Single file per firmware release
- Simple 'one click' firmware update
- Compact size (achieved by efficient compression of the firmware images)

- No installation required

When executed, the UPMF:

- Extracts its content into a temporary location
- Scans the locally installed NVIDIA devices firmware versions
- Performs firmware updates if needed
- Cleans up temporary files

UPMF Generation Flow

The mlx_fwsfx_gen tool is used for OEMs that wish to create their own UPMFs that contain their own customized firmware images.

To install the mlx_fwsfx_gen tool, the installation script should be run with the "--oem" command line option.

mlx_fwsfx_gen Usage

This tool packs the firmware images provided in the input directory and the mlxfwmanager update tool into a single standalone self-extracting executable.

The UPMF generation is supported on Linux and Windows. Being an executable file, the UPMF should be prepared for Linux and Windows separately.

Usage:

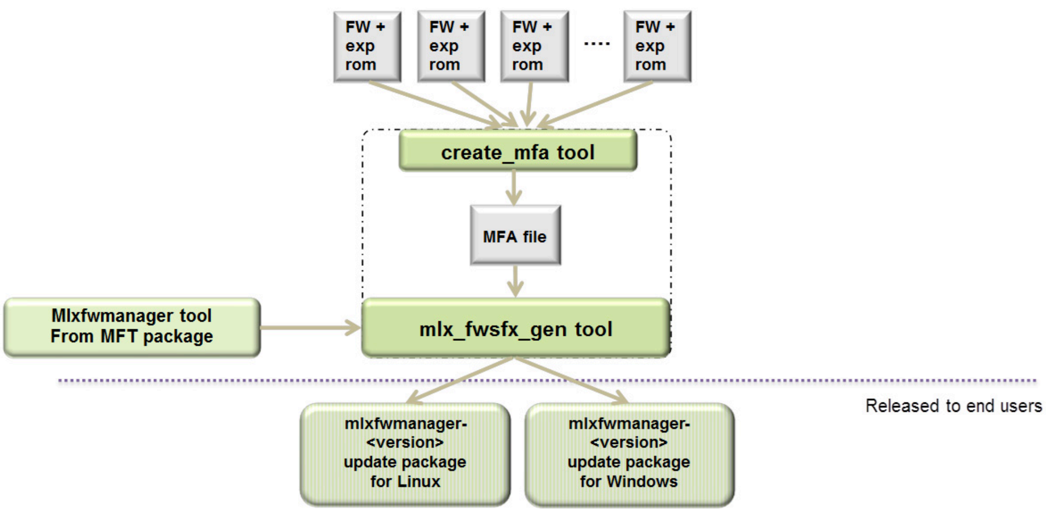
```
# mlx_fwsfx_gen --source-dir <FW images directory> --out-dir
<output directory> [--sfx-name <sfx file name>] [--phy-support --
phy-img <phy-img>][--extra-args <args>]
```

where:

--source-dir	Directory containing NVIDIA firmware images to be included in the package. This option may be used more than once to specify more than one source directory.

--out-dir	Specifies the output directory.
--certificate	SSL certificate.
--phy-support	Generate extractor with mlxphyburn support.
--phy-img	PHY firmware image.
--sfx-name	The self-extracting executable filename. The default name is mlxfwmanager-YYYYMMDD-<build number>, where build number is the previous maximum build number existing in the output directory incremented by one.
--extra-args	Extra args passed to mlxfwmanager default arguments. In the case of multiple args, the args are separated by commas. For example: [--extra-args --ret-lvim,--online]

UPMF Package Generation Flow



UPMF Generation Example

The below example packs 3 firmware binaries (named fw-ConnectX-3-1.bin, fw-ConnectX-3-2.bin, fw-ConnectX-3-3.bin) located in the directory '/tmp/fw-ConnectX-3-dir/' into a Linux UPMF package named /tmp/mlxfwmanager-20171004-1.

```
mlx_fwsfx_gen --source-dir /tmp/fw-ConnectX-3Pro-dir/ --out-dir /tmp/
```

```
Package name: /tmp/mlxfwmanager-20171004-1
Contents:
Source dirs: /tmp/fw-ConnectX-3Pro-dir
Adding file: /etc/mft/ca-bundle.crt
sfx_stub file: /usr/bin/mlx_sfx_stub
Creating intermediate MFA archive from binary files:
4779-314A-X00_Ax~ATT1090111023.bin
Huawei_TD70VMTA_VA_CX3Pro_2P_40G_Ax~HUA0020010017.bin
Inventec_U50_CX3Pro_10GE_A1~INV0010110023.bin
MCX342A-XCQ_Ax~MT_1680116023.bin
MCX353A-FCC_Ax~MT_1100111019.bin
mfa tool: /usr/bin/mlx_mfa_gen
mfa cmd: /usr/bin/mlx_mfa_gen -p /tmp/OMs1D5PvHq/srcs.mfa -s
/tmp/fw-ConnectX-3Pro-dir
Adding bins from /tmp/fw-ConnectX-3Pro-dir
Files copied: 5
Querying images ...
Files queried: 5
Compressing ... (this may take a minute)
Archive: /tmp/OMs1D5PvHq/srcs.mfa
Total time: 0m3s
Adding file: /tmp/OMs1D5PvHq/srcs.mfa
Adding file: /usr/bin/mlxfwmanager
Creating zip /tmp/OMs1D5PvHq/zippackage.zip
adding: srcs.mfa (deflated 0%)
adding: mlxfwmanager (deflated 52%)
adding: ca-bundle.crt (deflated 45%)
sfx auto-run command:
mlxfwmanager -u --log-on-update --ssl-certificate %ca-bundle.crt%
%current-dir% %argv%
Log name: /tmp/mlxfwmanager-20171004-1.log
```

UPMF Generation with PHY Binary Example

The below example packs 3 firmware binaries (named fw-ConnectX-3-1.bin, fw-ConnectX-3-2.bin, fw-ConnectX-3-3.bin) located in the directory '/tmp/fw-ConnectX-3-dir/' and a

PHY image '/tmp/Firmware_1.37.10_N32722.cld' into a Linux UPMF package named /tmp/mlxfwmanager-20141126-2.

```
mlx_fwsfx_gen --source-dir /tmp/fw-ConnectX-3-dir --out-dir /tmp
--phy-support --phy-img /tmp/
Firmware_1.37.10_N32722.cld

Creating /tmp/C04TldeQHr/phy_mfa direcotry
Package name: /tmp/mlxfwmanager-20141126-2
Contents:
Source dirs: /tmp/fw-ConnectX-3-dir
Adding file: /etc/mft/ca-bundle.crt
sfx_stub file: /usr/bin/mlx_sfx_stub
Creating intermediate MFA archive from binary files:
fw-ConnectX-3-1.bin
fw-ConnectX-3-2.bin
fw-ConnectX-3-3.bin
mfa tool: /usr/bin/mlx_mfa_gen
mfa cmd: /usr/bin/mlx_mfa_gen -p /tmp/YaH5BAoQ8q/srcs.mfa -s
/tmp/fw-ConnectX-3-dir
Adding bins from /tmp/fw-ConnectX-3-dir

Files copied: 3

Querying images ...
Files queried: 3
Compressing ... (this may take a minute)

Archive: /tmp/YaH5BAoQ8q/srcs.mfa
Total time: 0m1s
Adding file: /tmp/YaH5BAoQ8q/srcs.mfa
Adding file: /usr/bin/mlxfwmanager
Copying /tmp/Firmware_1.37.10_N32722.cld to
/tmp/C04TldeQHr/phy_mfa
Adding file: /tmp/Firmware_1.37.10_N32722.cld
```

```
Adding file: /usr/bin/mlxphyburn
Creating zip /tmp/YaH5BAoQ8q/zippackage.zip
adding: srcs.mfa (deflated 0%)
adding: ca-bundle.crt (deflated 45%)
adding: phy_mfa/ (stored 0%)
adding: phy_mfa/Firmware_1.37.10_N32722.cld (deflated 44%)
adding: mlxfwmanager (deflated 57%)
adding: mlxphyburn (deflated 60%)
sfx auto-run command:
mlxfwmanager -u --log-on-update --ssl-certificate %ca-bundle.crt%
%current-dir% %argv%
mlxphyburn auto-run command:
mlxphyburn %device% -i ./phy_mfa/Firmware_1.37.10_N32722.cld b
Log name: /tmp/mlxfwmanager-20141126-2.log
```

Updating Firmware Using an UPMF

Updating the firmware is done by simply executing the UPMF. Most of the command line options of the mlxfwmanager tool apply also for the UPMF.

For further detail, please refer to mlxfwreset - Loading Firmware on 5th Generation Devices Tool.

Some of the most commonly used command line options are:

--force	Force firmware update even if the firmware in the UPMF is not newer than the one on the device.
--yes	Non-interactive mode - assume 'yes' to all questions.

In addition to the mlxfwmanager tool command line options, the UPMF has 2 additional options:

Additional UPMF self extractor options:

--sfx-extract-dir <dir>	Use <dir> for temporary files during execution

--sfx-no-pause	Do not wait for user keypress after completion. Note: This flag is used in Windows OSs.

Extraction Example

```
# mlxfwmanager-20130717-1 --sfx-extract-dir ./mydir --sfx-extract-only
Extracting to mydir/mlxfwmanager-20130717-1 ... Done
```

Run the firmware update command:

```
# ./mydir/mlxfwmanager-20130717-1/mlxfwmanager -u
```

mlxarchive – Binary Files Compression Tool

Note

mlxarchive is not installed by default, and requires installing MFT with the `--oem` option.

The mlxarchive tool allows the user to create a file with the MFA2 extension. The new file contains several binary files of a given firmware for different adapter cards.

mlxarchive accepts the following attributes as its input:

- --bins-dir - The path to a folder with the binary files that will be included in the MFA2 file
- --version - The MFA2 file’s version

- --out-file - The output of the mlxarchive file (MFA2 file)
- -m|--mfa2-file mfa2_file - MFA2 file to parse

Example:

```
mlxarchive --bins-dir /full/path/to/bin/directory/ --version 1.1.1
--out-file out.mfa2 mlxarchive --mfa2-file out.mfa2
Creation Time   : 2019-09-18 08:35:43
Devices 2
PSID     : <…>
Num of Images    1
Index    0
Version : 10.16.1200
Date     : 2019-09-18 08:35:43 PSID   : <…>
Num of Images    1
Index    1
Version : 10.16.1200
Date     : 2019-09-18 08:35:43
```

mlxarchive Synopsis

```
[--help] [--version version] [--out-file out_file] [--bins-dir
bins_dir] [-m|--mfa2-file mfa2_file]
```

where:

--help	Shows the help message and exit
--version version	MFA2's version in the following format: x.x.x
--out-file out_file	The output file
-bins-dir bins_dir	The directory with the binaries files
-m|--mfa2-file mfa2_file	Mfa2 file to parse

Info

The .mfa2 file can be used with ethtool to burn adapter cards firmware. The procedure is described in Updating Firmware Using ethtool/devlink and .mfa2 File section.

mlxconfig – Changing Device Configuration Tool

The mlxconfig tool allows the user to change some of the device configurations without reburning the firmware. The configuration is kept after reset.

By default, mlxconfig shows the configurations that will be loaded in the next boot. For 5th generation devices, it is also possible to query the default configurations and the configurations that are used by the current running firmware.

Tool Requirements

- mst driver loaded
- OFED/WinOF driver to be installed and enabled
- Access to the device through the PCI interface (pciconf/pci_cr)
- Changing device configurations enabled

Note

For changes after a successful configuration to take effect, reboot the system.

mlxconfig Synopsis

```
# mlxconfig [Options] <commands> [Parameters]
```

where:

-d|--dev <device>	Performs operation for a specified mst device.
-b|--db <filename>	Use a specific database file.
-f|--file <conf.file>	Raw configuration file.
-h|--help	Displays help message.
-v|--version	Displays version info.
-e|--enable_verbosity	Show default and current configurations. Note: For 5th generation (Group II) devices, the --enable_verbosity option works with ConnectX-4 firmware v12.14.0016 and above for querying the default configurations, and with ConnectX-4 firmware v12.17.1010 and above for querying the current configurations.
-j|--json_format <file>	Save the query output to file in JSON format, only usable with Query command.
-y|--yes	Answers yes in prompt.
--with_default	Set command will fill any missing parameters with default values. If the final configurations matches the current, no set will be done.
-a|--all_attrs	Show all attributes in the XML template.
--host_id	Specify host id for Per-Host TLV (class 3).
--pf_index	Specify PF index for Per-Host TLV (class 3).
-p|--private_key	pem file for private key.
-u|--key_uuid	keypair uuid.
-eng|--openssl_engine	OpenSSL engine name.
-k|--open_ssl_key_id	OpenSSL key identifier
--aws_hsm	Sign in 3S environment.
-l|--private_key_label	Private key label to use for 3S HSM sign.
-t|--device_type	Specify the device type.

<switch/hca/linkx>	
--i2c_secondary	Specify I2C secondary address.
-s|--session_id	Specify the session id for token keep alive session.
-st|--session_time	Specify session time for token keep alive session.
-tkn|--token_type	Specify token type.
--sign_algorithm	Specify a signature algorithm from the following: RSA4k, RSA3k or ECDSA256.
--nested_token	Include challenge response for ArcusE.
clear_semaphore	Clear the tool's semaphore.
i[show_confs]	Display information about all configurations.
q[uery]	Queries the supported configurations. Note: Query command will query a single device if a device is specified. Otherwise, it will query all devices on the machine.
r[eset]	Resets configurations to their default value.
s[et]	Sets configurations to a specific device.
set_raw	Sets raw configuration file (5th generation/Group II devices only).
get_raw	Gets raw configuration file (5th generation/Group II devices only).
backup	Backs up configurations to a file (only 5th generation (Group II) devices). Use set_raw command to restore file.
gen_tlvs_file	Generate a List of all TLVs. TLVs output file name must be specified.
g[en_xml_template]	Generate an XML template. TLVs input file name and XML output file name must be specified.
xml2raw	Generate a Raw file from an XML file. XML input file name and raw output file name must be specified.
raw2xml	Generate an XML file from a Raw file. raw input file name and XML output file name must be specified.
xml2bin	Generate binary configuration dump file from XML file. XML input file name and bin output file name must be specified.
create_conf	Generate configuration file from XML file. XML input file name and bin output file name must be specified.

apply	Apply a configuration file, that was created with create_conf command. bin input file name must be specified.
challenge_request	Send a token challenge request to the device. Token type must be specified.
remote_token_keep_alive	Start a remote token session for a specified time. session id must be specified.
token_supported	Query which tokens are supported.
query_token_session	Query the status of a token session.
end_token_session	End an active token session.

Bifurcation Configuration

Before working with the Lego Softbank CG1 and Lego C2 flavors, make sure that the device is set to NIC mode (CPU as a RC). Please use the following commands, as specified in the "CPU as RC" column.

Lego Configuration

Configuration of Lego Systems	CPU as RC	DPU as RC

Configuration to bifurcation mode	To configure a BlueField-3 card as a "CPU is RC" of NVMe (DPU as PCIe switch):
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y reset`
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y s PCI_SWITCH0_UPSTRAEM_PORT_BUS=0`
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y s PCI_SWITCH0_UPSTRAEM_PORT_PEX=0`
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y s PCI_BUS00_HIERARCHY_TYPE=1`
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y s PCI_BUS00_WIDTH=5`
		To configure a BlueField-3 card "DPU is RC" of NVMe (using SN towards the CPU):
		`mlxconfig -d /dev/mst/mt41692_pcicon reset`
		`mlxconfig -d /dev/mst/mt41692_pcicon s PCI_BUS00_HIERARCHY_T`
		`mlxconfig -d /dev/mst/mt41692_pcicon s PCI_BUS00_WIDTH=5`
		`mlxconfig -d /dev/mst/mt41692_pcicon s PCI_BUS00_SPEED=4`

Configuration of Lego Systems	CPU as RC	DPU as RC
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y s PCI_BUS00_SPEED=4`	`mlxconfig -d /dev/mst/mt41692_picon s PCI_BUS10_HIERARCHY_T`
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y s PCI_BUS10_HIERARCHY_TYPE=1`	`mlxconfig -d /dev/mst/mt41692_picon s PCI_BUS10_WIDTH=3`
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y s PCI_BUS10_WIDTH=3`	`mlxconfig -d /dev/mst/mt41692_picon s PCI_BUS10_SPEED=4`
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y s PCI_BUS10_SPEED=4`	`mlxconfig -d /dev/mst/mt41692_picon s PCI_BUS12_HIERARCHY_T`
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y s PCI_BUS12_HIERARCHY_TYPE=1`	`mlxconfig -d /dev/mst/mt41692_picon s PCI_BUS12_WIDTH=3`
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y s PCI_BUS12_WIDTH=3`	`mlxconfig -d /dev/mst/mt41692_picon s PCI_BUS12_SPEED=4`
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y s PCI_BUS12_SPEED=4`	`mlxconfig -d /dev/mst/mt41692_picon s PCI_BUS14_HIERARCHY_T`
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y s PCI_BUS14_HIERARCHY_TYPE=1`	`mlxconfig -d /dev/mst/mt41692_picon s PCI_BUS14_WIDTH=3`
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y s PCI_BUS14_WIDTH=3`	`mlxconfig -d /dev/mst/mt41692_picon s PCI_BUS14_SPEED=4`
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y s PCI_BUS14_SPEED=4`	`mlxconfig -d /dev/mst/mt41692_picon s PCI_BUS16_HIERARCHY_T`
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y s PCI_BUS16_HIERARCHY_TYPE=1`	`mlxconfig -d /dev/mst/mt41692_picon s PCI_BUS16_WIDTH=3`
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y s PCI_BUS16_WIDTH=3`	`mlxconfig -d /dev/mst/mt41692_picon s PCI_BUS16_SPEED=4`

Configuration of Lego Systems	CPU as RC	DPU as RC
	`mlxconfig -d /dev/mst/mt41692_pciconf0 -y s PCI_BUS16_SPEED=4`	
Configuring the card in NIC mode	`# mlxconfig -d /dev/mst/mt41692_pciconf0 s INTERNAL_CPU_OFFLOAD_ENGINE=1`	`# mlxconfig -d /dev/mst/mt41692_pcicon INTERNAL_CPU_OFFLOAD_EN`

Examples of mlxconfig Usage

Querying the Device Configuration

To query the device's configuration, use the following command line:

```
# mlxconfig -d <device> query
```

ConnectX-3 Example:

```
# mlxconfig -d /dev/mst/mt4099_pciconf0 q

Device type: ConnectX-3
PCI device: /dev/mst/
/dev/mst/mt4099_pciconf0

Device 1:
-----------
Configurations:            Next Boot
   SRIOV_EN                True(1)
   NUM_OF_VFS              16
   WOL_MAGIC_EN_P1         False(0)
```

```
WOL_MAGIC_EN_P2          False(0)
```

> **Note**
>
> N/A means that the device default configuration is set.

> **Note**
>
> For Array type parameters, the query command will not show a value for it. It will only show you the word "Array" and the range of the array.
>
> - For example: HOST_CHAINING_DESCRIPTORS Array[0..7]
> - To query the fifth element in the array, run: mlxconfig -d <device> query HOST_CHAINING_DESCRIPTORS[5]
> - To specify a range: mlxconfig -d <device> query HOST_CHAINING_DESCRIPTORS[3..7]
> - To set the fifth element in the array, run: mlxconfig -d <device> set HOST_CHAINING_DESCRIPTORS[5]=3
> - Or you can set value for more than one element: mlxconfig -d <device> set HOST_CHAINING_DESCRIPTORS[3..7]=3

ConnectX-4 Lx Example:

```
# mlxconfig -d /dev/mst/mt4117_pciconf0 --enable_verbosity q

Device #1:
----------
Device type: ConnectX4LX
```

```
PCI device: /dev/mst/mt4117_pciconf0


Configurations:       Default     Current       Next Boot
* NUM_OF_VFS          8           5             5
  SRIOV_EN            True(1)     True(1)       True(1)
The '*' shows parameters with next value different from
default/current value.
```

Setting Device Configuration

To set the device configuration, use the following command line:

```
# mlxconfig -d <device> set [Parameters....]
```

Example:

```
# mlxconfig -d /dev/mst/mt4099_pciconf0 set WOL_MAGIC_EN_P2=1
NUM_OF_VFS=24
Device type: ConnectX-3
PCI device: /dev/mst/mt4099_pciconf0
Configurations:       Next Boot       New
 NUM_OF_VFS           16              24
 WOL_MAGIC_EN_P2      False(0)        True(1)

Apply new Configuration?(y/n) [n]: y
Applying... Done!

-I- Please reboot the system to load new configurations.
```

Resetting Device Configuration to Default

To reset the device configuration to default, use the following command line:

```
# mlxconfig -d <device> reset
```

Example:

```
# mlxconfig -d /dev/mst/mt4099_pciconf0 reset
Reset configuration for device /dev/mst/mt4099_pciconf0? ? (y/n)
[n] : y
Applying... Done!


-I- Please power-cycle device to load new configurations.
>mlxconfig -d /dev/mst/mt4099_pciconf0 query


Device 1:
----------
Device type: ConnectX-3
PCI Device: /dev/mst/mt4099_pciconf0
Configurations:          Next Boot
  SRIOV_EN               True(1)
  NUM_OF_VFS             8
  WOL_MAGIC_EN_P1        False(0)
  WOL_MAGIC_EN_P2        False(0)
```

Setting Missing Parameters with Default Values

Set of commands that will fill any missing parameters with default values. If the final configurations matches the current, no set will be done.

```
# mlxconfig -d <device> --with_default set BOOT_LACP_DIS=False
```

Example:

```
# mlxconfig -d /dev/mst/mt41692_pciconf0 --with_default  set
BOOT_LACP_DIS=False
Found a difference, first difference on the list is
VIRTIO_BLK_EMULATION_NUM_MSIX from val 0 to 2
continue to reset and set

 Apply reset and set? (y/n) [n] : y

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

Device type:        BlueField3
Name:               900-9D3B6-00CV-AAB_Ax
Description:        Nvidia BlueField-3 BF3220 P-Series DPU
200GbE/NDR200 dual-port QSFP112; PCIe Gen5.0 x16 FHHL with x16
PCIe extension option; Crypto Enabled; SB Disabled 32GB on-board
DDR; integrated BMC; Tall Bracket; IPN QP
Device:             /dev/mst/mt41692_pciconf0

Configurations:                                         Next
Boot        New
        BOOT_LACP_DIS                              True(1)
False(0)
Applying... Done!
-I- Please reboot machine to load new configurations.
```

Configuring Per-Host TLV Parameters: Host ID and PF Index

Used to define the target host and physical function for Class 3 Per-Host TLV operations.

```
# mlxconfig -d /dev/mst/mt41692_pciconf0 --host_id 1 --pf_index 1
set ROCE_CONTROL=2
```

Example:

```
# mlxconfig -d /dev/mst/mt41692_pciconf0 --host_id 1 --pf_index 1
set ROCE_CONTROL=2

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

Device type:         BlueField3
Name:                900-9D3B6-00CV-AAB_Ax
Description:         Nvidia BlueField-3 BF3220 P-Series DPU
200GbE/NDR200 dual-port QSFP112; PCIe Gen5.0 x16 FHHL with x16
PCIe extension option; Crypto Enabled; SB Disabled 32GB on-board
DDR; integrated BMC; Tall Bracket; IPN QP
Device:              /dev/mst/mt41692_pciconf0

Configurations:                                          Next
Boot        New
        ROCE_CONTROL
ROCE_ENABLE(2)       ROCE_ENABLE(2)

 Apply new Configuration? (y/n) [n] : y
Applying... Done!
-I- Please reboot machine to load new configurations.
```

Using mlxconfig

Using mlxconifg with PCI Device in Bus Device Function (BDF) Format

In order to access device in BDF format via configuration cycles, use "pciconf-" as prefix to the device.

Example:

```
# mlxconfig -d 0000:86:00.0 q

Device #1:
----------
Device type:    ConnectX5
Name:           MCX556A-ECA_Ax
Description:    ConnectX-5 VPI adapter card; EDR IB (100Gb/s) and
100GbE; dual-port QSFP28; PCIe3.0 x16; tall bracket; ROHS R6
Device:         0000:86:00.0

Configurations:                                      Next Boot
         MEMIC_BAR_SIZE                              0
         MEMIC_SIZE_LIMIT                            _256KB(1)
         HOST_CHAINING_MODE                          DISABLED(0)
         HOST_CHAINING_CACHE_DISABLE                 False(0)
         HOST_CHAINING_DESCRIPTORS                   Array[0..7]
         HOST_CHAINING_TOTAL_BUFFER_SIZE             Array[0..7]
         FLEX_PARSER_PROFILE_ENABLE                  0
         FLEX_IPV4_OVER_VXLAN_PORT                   0
         ROCE_NEXT_PROTOCOL                          254
         ESWITCH_HAIRPIN_DESCRIPTORS                 Array[0..7]
         ESWITCH_HAIRPIN_TOT_BUFFER_SIZE             Array[0..7]
         PF_BAR2_SIZE                                0
         NON_PREFETCHABLE_PF_BAR                     False(0)
         VF_VPD_ENABLE                               False(0)
         STRICT_VF_MSIX_NUM                          False(0)
         VF_NODNIC_ENABLE                            False(0)
```

```
          NUM_OF_VFS                                    0
          PARTIAL_WRITE_CACHE_MODE                      DEVICE_DEFAULT(0)
          PF_BAR2_ENABLE                                False(0)
          SRIOV_EN                                      False(0)
          PF_LOG_BAR_SIZE                               5
          .............
```

Using mlxconfig to Set IB/ETH Parameters

In order to set IB/ETH parameters through mlxconfig, use the following command line:

```
# mlxconfig -d <device> set [LINK_TYPE_P1=<link_type>]
[LINK_TYPE_P2=<link_type>]
```

Example: Configuring both ports as InfiniBand:

```
# mlxconfig -d /dev/mst/mt4119_pciconf0 set LINK_TYPE_P1=1
LINK_TYPE_P2=1

Device #1:
----------
Device type:    ConnectX5
Name:           MCX556A-ECA_Ax
Description:    ConnectX-5 VPI adapter card; EDR IB (100Gb/s) and
100GbE; dual-port QSFP28; PCIe3.0 x16; tall bracket; ROHS R6
Device:         /dev/mst/mt4119_pciconf0

Configurations:                                 Next Boot       New
         LINK_TYPE_P1                           ETH(2)
IB(1)
         LINK_TYPE_P2                           ETH(2)
IB(1)
```

```
Apply new Configuration? (y/n) [n] : y
Applying... Done!
-I- Please reboot machine to load new configurations.
```

Using mlxconfig to Set PHY Parameters

Setting Auto Negotiation

Auto negotiation modes is displayed at mlxconfig available configuration menu (Using <i> command).

In order to set non-volatile auto negotiation mode through mlxconfig, use the following command line:

```
# mlxconfig -d <device> set [PHY_AUTO_NEG_P1=
<auto_negotiation_mode>] [PHY_AUTO_NEG_P2=
<auto_negotiation_mode>]
```

Example: Configuring both ports with auto negotiation enabled:

```
# mlxconfig -d  /dev/mst/mt4129_pciconf0 set PHY_AUTO_NEG_P1=1
PHY_AUTO_NEG_P2=1

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

Device type:    ConnectX7
Name:           CX713106AE-HEA_QP1_Ax
Description:    NVIDIA ConnectX-7 HHHL adapter Card; 200GbE (default
mode) / NDR200 IB; Dual-port QSFP112; PCIe 5.0 x16 with x16 PCIe
extension option; Crypto Enabled; Secure Boot Capable; IPN for
internal use
Device:         /dev/mst/mt4129_pciconf0
```

```
Configurations:                                     Next Boot
New
        PHY_AUTO_NEG_P1
DEVICE_DEFAULT(0) AUTO_NEG_ENABLED(1)
        PHY_AUTO_NEG_P2
DEVICE_DEFAULT(0) AUTO_NEG_ENABLED(1)

 Apply new Configuration? (y/n) [n] : y
Applying... Done!
-I- Please reboot machine to load new configurations.
```

Setting Speed Rate Mask

Speed rate mask enables masking device's available speeds in all link operational modes.

Speed masking requires defining the speed rate mask and enabling speed override configuration.

In order to set non-volatile speed rate mask through mlxconfig, verify first that override mask is enabled using the following command:

```
# mlxconfig -d <device> set [PHY_RATE_MASK_OVERRIDE_P1=
<True|False>] [PHY_RATE_MASK_OVERRIDE_P2=<True|False>]
```

Example: Enabling both ports' speed mask override:

```
# mlxconfig -d  /dev/mst/mt4129_pciconf0 set
PHY_RATE_MASK_OVERRIDE_P1=1 PHY_RATE_MASK_OVERRIDE_P2=1

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

```
Device type:    ConnectX7
Name:           CX713106AE-HEA_QP1_Ax
Description:    NVIDIA ConnectX-7 HHHL adapter Card; 200GbE (default
mode) / NDR200 IB; Dual-port QSFP112; PCIe 5.0 x16 with x16 PCIe
extension option; Crypto Enabled; Secure Boot Capable; IPN for
internal use
Device:         /dev/mst/mt4129_pciconf0

Configurations:                                     Next Boot
New
        PHY_RATE_MASK_OVERRIDE_P1                  False(0)
True(1)
        PHY_RATE_MASK_OVERRIDE_P2                  False(0)
True(1)

 Apply new Configuration? (y/n) [n] : y
Applying... Done!
-I- Please reboot machine to load new configurations.
```

Configure the wanted speed mask using the following command:

```
# mlxconfig -d <device> set [PHY_RATE_MASK_P1=<rate_mask>]
[PHY_RATE_MASK_P2=<rate_mask>]
```

The rate mask mapping is displayed at mlxconfig available configuration menu (Using <i> command).

Example: Enabling both ports to support CAUI-4 / 100GBASE-CR4 only:

```
# mlxconfig -d  /dev/mst/mt4129_pciconf0 set
PHY_RATE_MASK_P1=0x200 PHY_RATE_MASK_P2=0x200

Device #1:
```

```
----------

Device type:    ConnectX7
Name:           CX713106AE-HEA_QP1_Ax
Description:    NVIDIA ConnectX-7 HHHL adapter Card; 200GbE (default
mode) / NDR200 IB; Dual-port QSFP112; PCIe 5.0 x16 with x16 PCIe
extension option; Crypto Enabled; Secure Boot Capable; IPN for
internal use
Device:         /dev/mst/mt4129_pciconf0

Configurations:                                     Next Boot
New
        PHY_RATE_MASK_P1                           0
0x200(512)
        PHY_RATE_MASK_P2                           0
0x200(512)

 Apply new Configuration? (y/n) [n] : y
Applying... Done!
-I- Please reboot machine to load new configurations.
```

Setting FEC Mode

Customizes FEC configuration of the port.

Detailed information on FEC modes is displayed at mlxconfig available configuration menu (Using *command).*

In order to set non-volatile FEC mode through mlxconfig, use the following command line:

```
# mlxconfig -d <device> set [PHY_FEC_OVERRIDE_P1=<fec_mode>]
[PHY_FEC_OVERRIDE_P2=<fec_mode>]
```

Example: Setting FEC mode disabled DME, RS FEC for 25G/50G/100G for both port:

```
# mlxconfig -d  /dev/mst/mt4129_pciconf0 set
PHY_FEC_OVERRIDE_P1=0x2 PHY_FEC_OVERRIDE_P2=0x2

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

Device type:    ConnectX7
Name:           CX713106AE-HEA_QP1_Ax
Description:    NVIDIA ConnectX-7 HHHL adapter Card; 200GbE (default
mode) / NDR200 IB; Dual-port QSFP112; PCIe 5.0 x16 with x16 PCIe
extension option; Crypto Enabled; Secure Boot Capable; IPN for
internal use
Device:         /dev/mst/mt4129_pciconf0

Configurations:                                     Next Boot
New
        PHY_FEC_OVERRIDE_P1
DEVICE_DEFAULT(0) MODE_2(2)
        PHY_FEC_OVERRIDE_P2
DEVICE_DEFAULT(0) MODE_2(2)

 Apply new Configuration? (y/n) [n] : y
Applying... Done!
-I- Please reboot machine to load new configurations.
```

Using mlxconfig to Set SR-IOV Parameters

In order to set SR-IOV parameters through mlxconfig, use the following command line:

```
# mlxconfig -d <device> set [SRIOV_EN=<0|1>] [NUM_OF_VFS=<NUM>]
```

Example: Turning on SR-IOV and enabling 8 Virtual Functions per Physical Function:

```
# mlxconfig -d /dev/mst/mt4115_pciconf0 set SRIOV_EN=1
NUM_OF_VFS=8

Device #1:
----------
Device type:    ConnectX4
PCI device:     /dev/mst/mt4115_pciconf0
Configurations:     Next Boot       New
  SRIOV_EN          0               1
  NUM_OF_VFS        0               8

Apply new Configuration? ? (y/n) [n] : y Applying... Done!
-I- Please reboot machine to load new configurations.
```

Using mlxconfig to Set Preboot Settings

For a full description of the preboot configurable parameters, refer to Supported Configurations and their Parameters under "Preboot Settings".

Example: Enable boot option ROM on port 1, set boot retries to 3 and set the boot protocol to PXE (on ConnectX-3 Pro cards only).

```
# mlxconfig -d /dev/mst/mt4103_pciconf0 set
BOOT_OPTION_ROM_EN_P1=1 BOOT_RETRY_CNT_P1=3
LEGACY_BOOT_PROTOCOL_P1=1

Device #1:
----------
Device type:   ConnectX3Pro
PCI device:    /dev/mst/mt4103_pciconf0
Configurations:            Next Boot      New
  BOOT_OPTION_ROM_EN_P1    False(0)       True(1)
  BOOT_RETRY_CNT_P1        0              3
  LEGACY_BOOT_PROTOCOL_P1  2              1
```

```
Apply new Configuration? ? (y/n) [n] : y
Applying... Done!
-I- Please reboot machine to load new configurations.
```

Example: Configure VLAN ID to 3 on port 2

```
# mlxconfig -d /dev/mst/mt4119_pciconf0 set BOOT_VLAN=3

Device #1:
----------
Device type:    ConnectX5
Name:           MCX556A-ECA_Ax
Description:    ConnectX-5 VPI adapter card; EDR IB (100Gb/s) and
100GbE; dual-port QSFP28; PCIe3.0 x16; tall bracket; ROHS R6
Device:         /dev/mst/mt4119_pciconf0

Configurations:                                      Next Boot       New
         BOOT_VLAN                                   1               3

Apply new Configuration? (y/n) [n] : y
Applying... Done!
-I- Please reboot machine to load new configurations.
```

Using mlxconfig to Split a Port in a Remotely Managed Switch

The break-out cable is a unique NVIDIA capability, where a single physical quad-lane HDR or NDR port is divided into 2 dual-lane ports. It maximizes the flexibility of the end user to use the NVIDIA switch with a combination of dual-lane and quad-lane interfaces according to the specific requirements of its network. All system ports may be split into 2-lane ports. Splitting a port changes the notation of that port -

- In HDR ports:

From x/y to x/y/z, with “x/y” indicating the previous notation of the port prior to the split, and “z” indicating the number of the resulting sub-physical port (1,2). Each sub-physical port is then handled as an individual port. For example, splitting port 5 into 2 lanes gives the following new ports: 1/5/1 & 1/5/2.

- In NDR ports:

  From x/y/z to x/y/z/i, with “x/y/z” indicating the previous notation of the port prior to the split, and “i” indicating the number of the resulting single-lane port (1,2). Each sub-physical port is then handled as an individual port. For example, splitting port 1/5/1 into 2 lanes results in ports 1/5/1/1 and 1/5/1/2.

To enable the port split, the following actions are required:

Step 1. Set the Split Mode in a Remotely Managed Switch.

```
# mlxconfig -d <device>set SPLIT_MODE=1
```

Example:

```
# mlxconfig -d /dev/mst/mt54000_pciconf0 set SPLIT_MODE=1

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

```
Device type:     Quantum
Name:            N/A
Description:     N/A
Device:          /dev/mst/mt54000_pciconf0
Configurations:    Next Boot                  New
  SPLIT_MODE       NO_SPLIT_SUPPORT(0)        SPLIT_2X(1)
```

To create a query for the Split Mode parameter using mlxconfig:

```
# mlxconfig -d <device> q SPLIT_MODE
```

Example:

```
# mlxconfig -d /dev/mst/mt54000_pciconf0 q SPLIT_MODE

Device #1:
----------
Device type:    Quantum
Name:           N/A
Description:    N/A
Device:         /dev/mst/mt54000_pciconf0
Configurations:   Next Boot
SPLIT_MODE        SPLIT_2X(1)
```

Step 2. Split a Port in a Remotely Managed Switch.

- To split a specific port for one or more ports of (1-64) using mlxconfig:

  ```
  # mlxconfig -d <device> set
  SPLIT_PORT[<port_num>/<port_range>]=1
  ```

> **Note**
>
> Please note that although the command, is “set SPLIT_PORT[33..64]”, it splits a specific port for one or more ports of the higher ports (33-40).
>
> Please note that the first port is set as 1, e.g., [1], [1..64].

- How to turn on the split port for the **first port** only:

```
mlxconfig -d /dev/mst/SW_MT54000_EVB-SX-1_L00_lid-0x0001 set
SPLIT_PORT[1]=1
```

- How to turn on the split port for the **first 32 ports** (range of ports):

```
mlxconfig -d /dev/mst/SW_MT54000_EVB-SX-1_L00_lid-0x0001 set
SPLIT_PORT[1..32]=1
```

- How to turn on the split port for the **last 8 ports**:

```
mlxconfig -d /dev/mst/SW_MT54000_EVB-SX-1_L00_lid-0x0001 set
SPLIT_PORT[57..64]=1
```

- How to turn off the split port for **port 40**:

```
mlxconfig -d /dev/mst/SW_MT54000_EVB-SX-1_L00_lid-0x0001 set
SPLIT_PORT[40]=0
```

- How to query the split port for the **first 32 ports**:

```
mlxconfig -d /dev/mst/SW_MT54000_EVB-SX-1_L00_lid-0x0001 query
SPLIT_PORT[1..32]
```

Step 3. Reboot the switch, or run:

```
flint -d <device> swreset
```

To disable the port split, the following actions are required:

Step 1. Disable the Split Ports in a Remotely Managed Switch:

- To unsplit a specific port for one or more ports (1-32) using mlxconfig:

```
# mlxconfig -d <device> set
SPLIT_PORT[<port_num>/<port_range>]=0
```

Step 2. Disable the Split Mode in a Remotely Managed Switch.

```
# mlxconfig -d <device> set SPLIT_MODE=0
```

mlxconfig Raw Configuration Files

mlxconfig allows applying raw configuration file for a pre-set configuration. Raw configuration files are intended for advanced users. This document does not cover the generation of such files.

Set the raw configuration file:

```
# mlxconfig -f ./tlv_file.conf -d /dev/mst/mt4115_pciconf1
set_raw

Raw TLV #1 Info:
Length: 0xc
Version: 0
OverrideEn: 1
Type: 0x00000080
Data: 0xa0000000 0xa0020010 0x00000000

Raw TLV #2 Info:
Length: 0x4
Version: 0
OverrideEn: 1
Type: 0x01020012
Data: 0x0000000b

Raw TLV #3 Info:
Length: 0x8
Version: 0
OverrideEn: 1
Type: 0x00000190
Data: 0x00000010 0x00007d00

Raw TLV #4 Info:
Length: 0x4
Version: 0
OverrideEn: 1
Type: 0x00000082
Data: 0x0000000c
Operation intended for advanced users.

Are you sure you want to apply raw TLV file? ? (y/n) [n] : y
Applying... Done!
```

```
-I- Please reboot machine to load new configurations.
```

> **Note**
>
> Never apply files from an unreliable source.

mlxconfig Commands

mlxconfig Backup Command

The backup command is used to save the current non-volatile configurations (TLV) in the device into a file in raw TLV syntax so it can be restored anytime using the set_raw command.

mlxconfig backup command allows backing up the all of the configurations which are related only to the PCI Physical Function associated with the given MST device. To back up all of the device configurations, perform the operation from every PCI Physical Function the device exposes. Restoring the configurations must be made from the matching PCI Physical Function.

> **Note**
>
> In a MultiHost environment, these operations are required to be executed per host.

```
# mlxconfig -d /dev/mst/mt4117_pciconf0 -f /tmp/backup.conf
backup
Collecting...
Saving output...
Done!
```

```
# cat /tmp/backup.conf
MLNX_RAW_TLV_FILE
% TLV Type: 0x00000400, Writer ID: ICMD MLXCONFIG(0x09), Writer Host
ID:
0x00 0x00000014 0x00000400 0x00000000 0x00000000 0x000e0000 0x001000f6 0x20160526 0x1
# mlxconfig -d /dev/mst/mt4117_pciconf0 -f /tmp/backup.conf
set_raw
Raw TLV #1 Info:
Length: 0x14
Version: 0
OverrideEn: 0
Type: 0x00000400
Data: 0x00000000 0x000e0000 0x001000f6 0x20160526 0x11250000
Operation intended for advanced users.
Are you sure you want to apply raw TLV file? ? (y/n) [n] : y
Applying... Done!
-I- Please reboot machine to load new configurations.
```

Generating an XML Template for the Configurations

Users can generate an XML file that contains a template for the configurations. The template describes the configurations and their parameters. No values are included in the template.

To generate such a template, run the gen_tlvs_file command. This command will generate a file containing a list of all supported configurations by mlxconfig, with a zero appearing in the end of each configuration. To choose a configuration, change the 0 to 1, then save the file and run the gen_xml_template command. An XML file containing the required configurations will be generated.

Example:

```
# mlxconfig gen_tlvs_file /tmp/confs.txt
Saving output...
Done!
# cat /tmp/confs.txt
```

```
nv_host_to_bmc        0
nv_kdnet_data         0
nv_fpga_data          0
nv_packet_pacing      0
nv_debug_mode         0
nv_global_pci_conf    0
```

In order to include the nv_kdnet_data configuration in the template, change the 0 to 1, as demonstrated in the following example.

Example:

```

nv_kdnet_data                 1

#mlxconfig gen_xml_template /tmp/confs.txt /tmp/template.xml
Saving output...
Done!

#cat /tmp/template.xml
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://www.mellanox.com/config">
<nv_kdnet_data>

              <!-- Legal Values: False/True -->
              <kdnet_en></kdnet_en>

</nv_kdnet_data>
</config>
```

Advance Options

Add the `-a` flag In order to allow advance options in the XML generated file. When using this flag, each TLV in the XML file has additional attributes that must be filled.

by default, all TLVs will be at MLNX priority. Other possible values are OEM and USER.

mlxconfig xml2raw Command

The xml2raw command is an easy way to generate a flawless raw configuration file that can be used in the set_raw command. The input for the command is an XML file that contains the data of the required configurations. To generate an XML file and fill it with the desired values, run the commands from Generating an XML Template for the Configurations, and then use the xml2raw command to generate a raw file.

Example:

```
# cat /tmp/template.xml
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://www.mellanox.com/config">
<nv_kdnet_data>

     <!-- Legal Values: False/True -->
     <kdnet_en>True</kdnet_en>

</nv_kdnet_data>
</config>

#mlxconfig xml2raw /tmp/template.xml /tmp/confs.raw
Saving output...
Done!

#cat /tmp/confs.raw
MLNX_RAW_TLV_FILE
0x03000004 0x00000085 0x00000000 0x80000000
```

mlxconfig xml2bin Command

The xml2bin command is an easy way to generate a binary file that contains a binary dump of configurations. The input for the command is an XML file that contains the data of the required configurations. To generate an XML file and fill it with the desired values, run the commands from Section 2.4.10, and then use the xml2bin command to generate a binary file.

Example:

```
#mlxconfig xml2bin /tmp/template.xml /tmp/confs.bin
Saving output...
Done!
# hexdump -C /tmp/confs.bin
00000000  80  00  00  00                                |....|
00000004
```

mlxconfig create_conf Command

The create_conf command assists in creating an NV configuration file that can be used for LifeCycle and Secure Firmware Updates purposes. The flow for creating a configuration file is the same as the flow for xml2bin. The user must provide the command with an XML file containing the required configurations and their values. The command can sign the configuration file, if the user provides a private key and UUID. The sign result will be appended to the end of the configuration file. If no private key and UUID are provided, the tool will compute an SHA256/SHA512 digest and append it to the file. The generated signature will be used by the firmware for authentication purposes.

Note

Currently the only supported NV configurations types are CS tokens, Debug tokens, BTC tokens, and MLNX ID which are used for Secure Firmware Updates. Additionally, it supports two RSA keys of 2048 or 4096 bits length for all tokens except BTC which supports only 4096 bits.

Note

The NV configurations files must have the applicable_to configuration.

Examples:

```
# mlxconfig create_conf --private_key privatekey.pem --key_uuid
"29ee36ee-13b7-11e7-83de-0cc47a6d39d2" /tmp/template.xml /tmp/nvconf.bin
Saving output...
Done!
```

```
# mlxconfig create_conf --private_key privatekey.pem --key_uuid
"29ee36ee-13b7-11e7-83de-0cc47a6d39d2" /tmp/template.xml /tmp/nvconf.bin --
openssl_engine pkcs11 --openssl_key_id
"pkcs11:serial=0123456789abcdef;token=My%20token%201;type=private;object=example_pkey;id=%12%34%
--key_uuid "e0129552-13ba-11e7-a990-0cc47a6d39d2" /tmp/template.xml
/tmp/nvconf.bin
Saving output...
Done!
```

mlxconfig apply Command

The apply command can be used to apply the NV configurations files to the firmware using the apply command. Only Firmware that supports applying configurations files can be used.

Example:

```
# mlxconfig -d /dev/mst/mt4115_pciconf0 apply /tmp/nvconf.bin
Applying...
Done!
```

MFT Supported Configurations and Parameters

Note

The list of MFT Supported Configurations and Parameters is available by running the `“mlxconfig -d <device> show_confs”` command.

Note

Before setting the number of VFs in SR-IOV, make sure your system can support that number of VFs. If your hardware and software cannot support that number, this may cause your system to cease working. Therefore, mlxconfig protects the user by making sure that when setting SR-IOV

parameters, for ConnectX-3 and ConnectX-3 Pro, the value of NUM_OF_VFS*PCI_BAR_SIZE[(1)] must not exceed 512. For 5th generation devices (Group II devices), however, the value is dependent on the firmware. Also, NUM_OF_VFS must not exceed the limit defined by the firmware (127 VFs upper bound). The same calculation applies to BAR size settings.

[(1)]. PCI_BAR_SIZE refers to the PCI BAR size per function, either physical or virtual.

Note

In case there were no server booting after enabling SR-IOV, please refer to Troubleshooting.

Note

Support was added to set some of the parameters in mlxconfig in textual values in addition to the numerical values that are still supported. For example: LINK_TYPE_P1 can be set as follows: LINK_TYPE_P1=ETH, instead of: LINK_TYPE_P1=2

Note that the textual values are case insensitive (either “True” or “true” are accepted).

flint – Firmware Burning Tool

The flint (Flash interface) utility performs the following functions:

- Burns a binary firmware image to the Flash device attached to an adapter or a switch device.
- Burns an Expansion ROM image to the Flash device attached to adapters.
- Queries for firmware attributes (version, GUIDs, UIDs, MACs, PSID, etc.)
- Enables executing various operations on the Flash memory from the command line (for debug/production).
- Disables/enables the access to the device’s hardware registers, and changes the key used for enabling. This feature is functional only if the burnt firmware supports it.

flint Synopsis

```
flint [OPTIONS] <command> [parameters...]
```

Tool Options

--allow_rom_change	Allows burning/removing a ROM to/from Firmware image when product version is present.
-banks <banks>	Set the number of attached flash devices (banks)
-blank_guids	Burn the image with blank GUIDs and MACs (where applicable). These values can be set later using the "sg" command (see details below). Commands affected: burn
-clear_semaphore	Force clear the flash semaphore on the device. No command is allowed when this flag is used. NOTE: May result in system instability or flash corruption if the device or another application is currently using the flash. Exercise caution.
--cpu_util <CPU_UTIL>	Use this flag to reduce CPU utilization while burning, Windows only. Legal values are from 1 (lowest CPU) to 5 (highest CPU)
-d[evice] <device>	Device flash is connected to. Commands affected: all
-dual_image	Make the burn process burn two images on flash (previously default algorithm). Current default failsafe burn process burns a single image (in alternating locations). Commands affected: burn
-flash_params <type,log2size,num_of_flashes>	Use the given parameters to access the flash instead of reading them from the flash. Supported parameters: • Type: The type of the flash, such as: M25PXxx, M25Pxx, SST25VFxx, W25QxxBV, W25Xxx, AT25DFxxx, S25FLXXXP • log2size: The log2 of the flash size • num_of_flashes: the number of the flashes connected to the device
--flashed_version	When specified, only flashed fw version is fetched Commands affected: query
-guid <GUID>	GUID base value. 4 GUIDs are automatically assigned to the following values:

	guid -> node GUID guid+1 -> port1guid+2 -> port2guid+3 -> system image GUID NOTE: port2 guid will be assigned even for a single port HCA - The HCA ignores this value. Commands affected: burn, sg
-guids <GUIDs...>	4 GUIDs must be specified here. The specified GUIDs are assigned to the following fields, repectively: node, port1, port2 and system image GUID. NOTE: port2 guid must be specified even for a single port HCA. The HCA ignores this value. It can be set to 0x0. Commands affected: burn, sg
-h[elp]	Prints this message and exits
-hh	Prints extended command help
--hmac_key <hmac_key>	Path to the file containing the HMAC key (For FS4 image only).
--hsm	Flag to use with sign command. Will use HSM HW for encryption operations.
-i[mage] <image>	Binary image file. Commands affected: burn, verify
--ignore_dev_data	Do not attempt to take device data sections from device (sections will be taken from the image. FS3 image only). Commands affected: burn
--key_uuid <uuid_file>	UUID matching the given private key to be used by the sign command
--key_uuid2 <uuid_file>	UUID matching the given private key to be used by the sign command
-log <log_file>	Prints the burning status to the specified log file
--low_cpu	When specified, cpu usage will be reduced. Run time might be increased Commands affected: query
-mac <MAC>1	MAC address base value. 2 MACs are automatically assigned to the following values: mac -> port1 mac+1 -> port2 Commands affected: burn, sg

-macs <MACs...>[1]	2 MACs must be specified here. The specified MACs are assigned to port1, port2, respectively. Commands affected: burn, sg NOTE: -mac/-macs flags are applicable only for NVIDIA Ethernet products.
-no	Non-interactive mode - assume answer "no" to all questions. Commands affected: all
-no_flash_verify	Do not verify each write on the flash.
--no_fw_ctrl	Do not attempt to work with the firmware Ctrl update commands.
-nofs	Burns image in a non failsafe manner.
--openssl_engine <engine name>	Name of the OpenSSL engine to be used by the sign/rsa_sign commands to work with the HSM hardware via OpenSSL API ⚠ **Warning** Please be aware that OpenSSL engine will be deprecated in the next release.
--openssl_key_id <key>	Key identification string to be used by the sign/rsa_sign commands to work with the HSM hardware via OpenSSL API
--output_file <string>	Output file name for exporting the public key from PEM/BIN.
-override_cache_replacement[2]	Allow accessing the flash even if the cache replacement mode is enabled. NOTE: This flag is often referred to as -ocr NOTE: This flag is intended for advanced users only. Running in this mode may cause the firmware to hang.
--private_key <key_file>	Path to PEM formatted private key to be used by the sign command
--private_key_label <string>	Flag to use with sign/import_hsm_key commands.

--private_key2 <key_file>	Path to PEM formatted private key to be used by the sign command
-qq	Run a quick query. When specified, flint will not perform full image integrity checks during the query operation. This may shorten execution time when running over slow interfaces (e.g., I2C, MTUSB-1). Commands affected: query
-s[ilent]	Do not print burn progress flyer. Commands affected: burn
-striped_image	Use this flag to indicate that the given image file is in a "striped image" format. Commands affected: query verify
-uid <UID>	5th Generation (Group II) devices only. Derive and set the device's base UID. GUIDs and MACs are derived from the given base UID according to NVIDIA Methodologies. Commands affected: burn, sg
-use_dev_rom	Save the ROM which exists in the device (FS3 and FS4 image only). Commands affected: burn
--use_fw	Access to flash using FW (ConnectX-3/ConnectX-3 Pro device only). Commands affected: all
-use_image_guids	Burn (guids/uids/macs) as appears in the given image. Commands affected: burn
-use_image_ps	Burn vsd as appears in the given image - do not keep existing VSD on flash. Commands affected: burn
-use_image_rom	Do not save the ROM which exists in the device. Commands affected: burn
--user_password <string>	Flag to use with HSM HW for encryption operations.
-v	Version info.
-vsd <string>	Write this string, of up to 208 characters, to VSD when burn.
-y[es]	Non interactive mode - assume answer "yes" to all questions. Commands affected: all

--linkx	Burn or query the cable device connected to the host.
--aws_hsm	Flag for the `rsa_sign` command
--cert_chain_index	Use this flag to specify the certificate location. The acceptable values are 0-7 (default - 0).
--i2c_secondary <address>	Change the I^2C secondary address.
--run_module_image	Use this flag to switch FW banks on the module. By default, the command is not performed.
--commit_module_image	Use this flag to switch committed bank on the module. By default, the command is not performed.

> **Note**
>
> The -mac and -macs options are applicable only to NVIDIA Ethernet adapter and switch devices.

> **Note**
>
> When accessing SwitchX via I^2C or PCI, the -override_cache_replacement flag must be set.

Command Parameters

The flint utility commands are:

Common Firmware Update and Query

b[urn]	Burn flash
q[uery] [full]	Query misc. flash/firmware characteristics, use "full" to get more information.
v[erify]	Verify entire flash

swreset	SW reset the target unmanaged switch device. This command is supported only in the In-Band access method.
sign_with_hmac	Sign image with HMAC.
component_type	Queries components on a given device via FW
skip_if_same	Skip burning if the firmware version matches the current version on the device

Expansion ROM Update:

brom <ROM-file>	Burn the specified ROM file on the flash.
drom	Remove the ROM section from the flash.
rrom <out-file>	Read the ROM section from the flash.
qrom	Query ROM in a given image.

Initial Burn, Production:

bb	Burn Block - Burns the given image as is. No checks are done. Note: The MFT 'bb' option is an advanced flag used ONLY for production flows.It is NOT recommend to use it, as it can cause unrecoverable firmware burning failures. **Note:** FwBurnBlock is not supported any longer in FS3 and up images.
sg [guids_num= <num> step_size=<size>] | [nocrc]	Set GUIDs.
set_vpd [vpd file]	Set read-only VPD (For FS3 image only).
smg [guids_num= <num> step_size=<size>]	Set manufacture GUIDs (For FS3 image only).
sv	Set the VSD.

Misc Firmware Image Operations

ri <out-file>	Read the fw image on the flash.
dc [out-file]	Dump Configuration: print fw configuration file for the

<table>
<tr><td></td><td>given image.</td></tr>
<tr><td>dh [out-file]</td><td>Dump Hash: print hash file for the given image.</td></tr>
<tr><td>checksum|cs</td><td>Perform MD5 checksum on firmware.</td></tr>
<tr><td>timestamp|ts <set|query|reset> [timestamp] [FW version]</td><td>Set/query/reset firmware timestamp.</td></tr>
<tr><td>cache_image|ci</td><td>Cache FW image (Windows only).</td></tr>
<tr><td>sign</td><td>Sign firmware image file</td></tr>
<tr><td>rsa_sign</td><td>Sign firmware image file with RSA</td></tr>
<tr><td>set_public_keys [public key binary file]</td><td>Set Public Keys (For FS3/FS4 image only).</td></tr>
<tr><td>set_forbidden_versions [forbidden versions]</td><td>Set Forbidden Versions (For FS3/FS4 image only).</td></tr>
<tr><td>import_hsm_key</td><td>This command allows to import the private and public key to the HSM HW.</td></tr>
<tr><td>export_public_key</td><td>This command extracts the public key from given BIN file or from PEM file.</td></tr>
</table>

Hardware Access Key:

<table>
<tr><td>set_key [key]</td><td>Set/Update the HW access key which is used to enable/disable access to HW.
The key can be provided in the command line or interactively typed after the command is given.NOTE: The new key is activated only after the device is reset.</td></tr>
<tr><td>hw_access <enable|disable> [key]</td><td>Enable/disable the access to the HW.
The key can be provided in the command line or interactively typed after the command is given.</td></tr>
</table>

Low Level Flash Operations:

hw query	Query HW info and flash attributes.
e[rase] <addr>	Erase sector
rw <addr>	Read one dword from flash
ww <addr> < data>	Write one dword to flash

wwne <addr>	Write one dword to flash without sector erase
wb <data-file> <addr>	Write a data block to flash
wbne <addr> <size> <data ...>	Write a data block to flash without sector erase
rb <addr> <size> [out-file]	Read a data block from flash

Return Values:

Value	Description
0	Successful completion
1	An error has occurred
7	For the `burn` command - The option of burning new firmware was not chosen by the user when prompted. Thus, the firmware burning process was aborted.

> **Note**
>
> The following commands are non-failsafe when performed on a 5th generation (Group II) device: `sg, smg, sv and set_vpd`.

> **Note**
>
> Manufacture GUIDs are similar to GUIDs. However, they are located in the protected area of the flash and set during production. By default, firmware will use GUIDs unless specified otherwise during production.

Burning a Firmware Image

The flint utility enables you to burn the Flash from a binary image.To burn the entire Flash from a raw binary image, use the following command line:

```
# flint -d <device> -i <fw-file> [-guid <GUID> | -guids <4 GUIDS>
| -mac <MAC> | -macs <2 MACs>] burn
```

where:

device	Device on which the flash is burned.
fw-file	Binary firmware file.
GUID(s)	Optional, for InfiniBand adapters and 4th generation (Group I) switches. One or four GUIDs. • If 4 GUIDS are provided (-guids flag), they will be assigned as node, Port 1, Port 2 and sys- tem image GUIDs, respectively. • If only one GUID is provided (-guid flag), it will be assigned as node GUID. Its values +1, +2 and +3 will be assigned as Port 1, Port 2 and system image GUID, respectively. • If no -guid/-guids flag is provided, the current GUIDs will be preserved on the device. NOTE: For 4th generation (Group I), four GUIDs must be specified but Ports 1 and 2 GUIDs are ignored and should be set to 0. NOTE: A GUID is a 16-digit hexadecimal number. If less than 16 digits are provided, leading zeros will be inserted.
MAC(s)	Optional, for Ethernet and InfiniBand adapters and switches. • If 2 MACs are provided (-macs flag), they will be assigned to Port 1 and Port 2, respectively. • If only one MAC is provided (-mac flag), it will be assigned to Port 1; MAC+1 will be assigned to Port 2. • If no -mac/-macs flag is provided, the current LIDs will be preserved on the device. NOTE: A MAC is a 12-digit hexadecimal number. If less than 12 digits are provided, leading zeros will be inserted.

To burn a firmware image:

1. Update the firmware on the device, keeping the current GUIDs and VSD. (Note: This is the common way to use the tool.)

```
# flint -d /dev/mst/mt4131_pciconf0 -i fw_900-9X81Q-00CN-
STA_Ax~MT_0000001223.bin burn
```

2. Update the firmware on the device, specifying the GUIDs to burn.

```
# flint -d /dev/mst/mt4131_pciconf0 -i fw_900-9X81Q-00CN-
STA_Ax~MT_0000001223.bin -guid 1234567deadbeef burn
```

3. Update the firmware on the device, specifying the MACs to burn.

```
# flint -d /dev/mst/mt4131_pciconf0 -i fw_900-9X81Q-00CN-
STA_Ax~MT_0000001223.bin -mac 1234567deadbeef burn
```

4. Burn the image on a blank Flash device. This means that no GUIDs are currently burnt on the device, therefore they must be supplied (with -guid/-guids) by the burning command. Moreover, the burn process cannot be failsafe when burning a blank Flash, therefore the -nofs flag must be specified.

```
burn# flint -d /dev/mst/mt4131_pciconf0 -i fw_900-9X81Q-00CN-
STA_Ax~MT_0000001223.bin -nofs -guid 12345678 burn
```

5. Read FW from the device and save it as an image file.

```
# flint -d /dev/mst/mt4131_pciconf0 ri Flash_Image_Copy.bin
```

6. MT58100 SwitchX switch:

   Burn the image on a blank Flash device. Meaning, no GUIDs/MACs are currently burnt on the device, therefore they must be supplied (with -guid/-guids and -mac/-

macs) by the burning command. Moreover, the burn process cannot be failsafe when burning a blank Flash, therefore the -nofs flag must be specified.

```
# flint -d /dev/mst/mtusb-1 -i /tmp/fw-sx.bin -nofs -guids
000002c900002100 0 0 000002c900002100 -macs 0002c9002100
0002c9002101 b
```

7. MT58100 SwitchX switch inband firmware update:

```
# flint -d lid-0x18 -i /tmp/fw-sx.bin b
```

Burning the MFA2 Images

Burning the MFA2 images enables the user to extract (i.e. unzip) 4MB images from MFA2 archive that matches the device type and device PSID. If there are more than one matching images, the user may use the `--latest_fw` flag and burn the latest firmware, or choose the required image from the user menu.

Note

The device flash **MUST** have all relevant device information (signatures, PSID, VPD, DEV_INFO, MFG_INFO, etc.) valid since MFA2 format does not have that information and without the burn process will fail.

```
flint -d <device> -i <mfa2 file> --psid <PSID string>
(optionally) --latest_fw (optionally) –silent
(optionally) b (or burn)
```

- **Burning the MFA2 Images when the Device Includes a Valid Image**

In this scenario, the user ***may*** (optional) provide a `“-psid”` flag and extract from the MFA2 archive the image that matches this flag, and this way actually change the PSID on the device.

- **Burning the MFA2 Images when in Live Fish Mode**

  In this scenario, the user ***must*** provide a `“-psid”` flag and extract from the MFA2 archive the image that matches this flag, and this way actually change the PSID on the device.

Burning PLDM Images

```
flint -d <device> -i <pldm_package> --component_type FW -y b
```

Options:

--component_type	Component type to burn from the package of the supported components (currently only FW)

Cable Firmware Update (In-Field-Firmware-Update)

Note

This capability is supported only in NVIDIA Quantum switch systems and hosts with NVIDIA ConnectX-6 adapter cards.

The In-Field-Firmware-Update (IFFU) tool works via the switches/NICs in the datacenters and is intended for remote control. The tool is used to update cables transceivers' firmware.

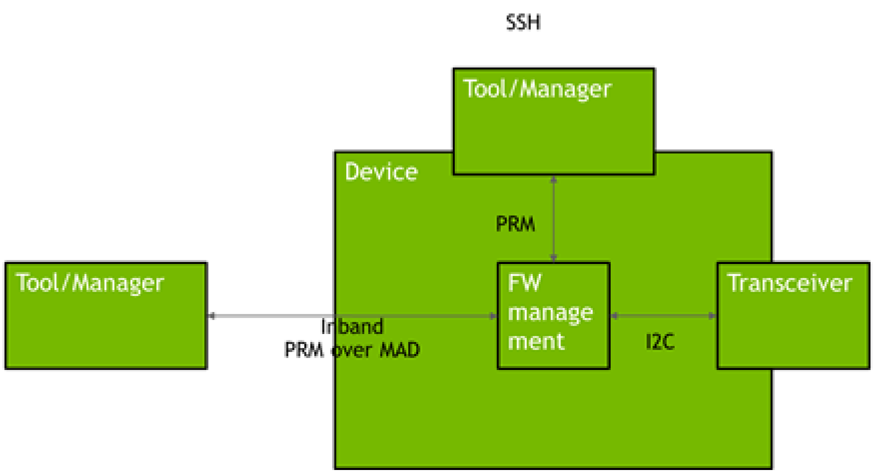
Optical Cables and Transceivers are active network components which run firmware, and as any component running firmware, the ability to update firmware is mandatory. Transceiver firmware update is a system flow which requires the following elements:

- Tool/Manager which will perform the firmware update
- Switch/NIC firmware management used as a middleman between the Manager and the cable transceiver

- Transceiver firmware: target for upgrade

The figure below shows the tool/manager which runs on a remotely controlled host or (in case of managed switches) on a switch, shown as ‘Device’.

The manager can query the transceivers type and the current running firmware to understand if an update is required. When an update is required, the manager can apply set of commands that will send the remote host device a new firmware images for the specific transceiver(s) and activate a firmware update flow. The set of commands is defined with low level primitives to support full flexibility for the user. High level script can be applied on top of the manager and allow system wide update.



> **Note**
>
> The update of modules/AOCs connected to switches is done over InfiniBand (inband) PRM registries. Whereas, the update of modules connected to NICs is done over MCC (RegAccess) on the host.
>
> Inband connection implies that unmanaged switches like QM8790 support IFFU.
>
> Each device (NIC, Switch) can update only the modules connected directly to it, not the far end. Updating the far end transceiver/end of the AOC requires the same operation to be done at the far end switch(es).

The Tool/Manager host must have MST rev. 4.16.00 or later installed.

Remote control from outside the cluster (data center) requires access to the host being used as Tool/Manager. When the cluster has many switches, multiple hosts may be engaged in the upgrade process. The host(s) can be remotely controlled via VNC access.

Firmware Burning Across a Cluster (Data Center)

The IFFU function described below works on one switch. Cluster-wide firmware updating is done by use of a script which initiates the update procedure in multiple switches in parallel by initiating an instance of the flint command for each switch. In large clusters the script can be executed on multiple hosts, each handling a different part of the cluster.

Cable Burn Command

```
# flint -d <device> --linkx <flags> <commands>
```

where:

Flags:

<device>	The name of the target switch (one only).
--downstream_device_id_start_index <downstream_device_id_start_index>	The port number of the first LinkX cable/transceiver. (min. port number = 1)
--num_of_downstream_devices <num_of_downstream_devices>	the number of cables/transceivers to burn. They are burnt sequentially.
--linkx_auto_update	Use this flag to burn all supported cables/transceivers connected to the switch.
--download_transfer	Use this flag to perform download and transfer of all cable data for cables. Download and transfer are not performed by default. This flag is only relevant for cable components.
--activate	Use this flag to apply the activation of the new firmware in the updated devices. Activation is not performed by default.

--activate_delay_sec <timeout in seconds>	Use this flag to activate all cable devices connected to host with delay, acceptable values are between 0 and 255 (default - 1, immediately). Important: 'activate' flag must be set. This flag is relevant only for cable components.
--i <image>	‘i’ indicates ‘binary Image’ followed by the path and file name of the bin file to download into the cable/transceiver.
--downstream_device_ids <list of ports>	Use this flag to specify the LNKX ports to perform query. List must be only comma-separated numbers, without spaces

Commands:

b[urn]	Burn flash
q[uery]	Query misc. flash/firmware characteristics.

Updating the Firmware

Burning a firmware cable transceiver connected to the host (NIC or switch) is done using the "flint" tool. To do so, the user should use the "–linkx" flag.

Firmware can be burnt in follow one of the methods:

- **Burn with Auto-update:**

1. Transfer the data from the host.

```
# flint -d <device> --linkx --linkx_auto_update --
download_transfer -i <image> b
```

Example:

```
# flint -d lid-2 --linkx --linkx_auto_update --
download_transfer -i image.bin b
```

2. Activate the firmware.

```
# flint -d <device> --linkx --linkx_auto_update —-activate b
```

Note

The flint "--activate" flag behavior is changed to include a minimal delay of 1 second to avoid disconnections if the connected port is being activated. To use the "legacy" activation flow, use the "--activate_delay_sec 0" command.

Example:

```
# flint -d lid-2 --linkx --linkx_auto_update --activate b
```

Activate with delay Example:

```
# flint -d lid-2 --linkx --linkx_auto_update --activate --
activate_delay_sec 10 b
```

Transfer and Activate Example:

```
# flint -d lid-2 --linkx --linkx_auto_update --
download_transfer --activate -i image.bin b
```

⚠ **Warning**

Burning all cables in an unmanaged switch in one operation is risky. If the cables do not link up after the update, you lose connection to the switch – permanently.

Burn half of the cables, check that they come up after burning, then burn the other half.

- **Burning multiple cables in the switch using the 'Range':**

1. Transfer the data from the host.

```
# flint -d <device> --linkx --
downstream_device_id_start_index <index> --
num_of_downstream_devices <number> --download_transfer -i
<image> b
```

2. Activate the firmware.

```
# flint -d <device> --linkx --
downstream_device_id_start_index <index> --
num_of_downstream_devices <number> --activate b
```

Example of Download Transfer with Activation, range indices is 10 to 16:

```
# flint -d lid-2 --linkx --downstream_device_id_start_index
10 --num_of_downstream_devices 6 download_transfer --activate
-i image.bin b
```

This will update 6 AOCs/Transceivers starting from port 10, i.e. all ports in the range 10...15.

> **Note**
>
> You cannot 'overburn' the same firmware version into a transceiver/AOC as the one already installed. This is to prevent wasting time re-burning transceivers in a large cluster. If you try to burn the existing FW version, the command responds:
>
> `Cable burn failed, error is LinkX downstream transfer failed for device index i`

Example of successful update of 1 AOC:

```
-I- Downloading FW ...
FSMST_INITIALIZE -   OK
Writing COMPID_LINKX component -   OK
FSMST_LOCKED -   OK
FSMST_DOWNSTREAM_DEVICE_TRANSFER -   OK
FSMST_LOCKED -   OK
Please wait while activating the transceiver(s) FW ...
FSMST_ACTIVATE -   OK..]
-I- Cable burn finished successfully.
```

> **Note**

Downloading and burning takes approx. 1½ minute + activation ½ minute for one cable. The time for multiple cables depends on which ports they are plugged into.

Cable Burn Command Running CMIS Firmware Upgrade Flow for Supported Cables

The flint tool is able to burn firmware packages on CMIS compliant cables that support the CDB firmware update procedure.

```
# flint --device <mst cable device> [--image <image>] [flags]
burn
```

Where:

Where:

--module_password	Optional, module password to enable locked operations.
-- module_vendor_data	Optional, path to vendor data file in case it is not a part of the firmware image file.
--activate	Optional, run and commit the burned image. Use without the “--image" flag to try to perform run and commit commands if possible.

Querying Firmware Version from an Image

Querying a cable image for firmware version is done using the "flint" tool.

```
# flint -i <fw file> q
```

Querying Vendor Specific Firmware Information from a NVIDIA AOC / Transceiver

Querying a firmware cable transceiver is done using the "flint" tool.

> **Note**
>
> In case the Vendor Specific query command is not support by the firmware, it will run the CMIS standard query implemented by the firmware.

```
# flint -d <cable device> q
```

Querying Firmware Information from an AOC / Transceiver

Querying a firmware cable transceiver connected to the host (NIC or switch) is done using the "flint" tool. To do so, the user should use the `"-linkx"` flag.

```
# flint -d <device> --linkx --downstream_device_ids <ids> [--
output_file <file_name>] q
```

Query ports 1,2,5 Example:

```
# flint -d <device> --linkx --downstream_device_ids 1,2,5 q
```

The system responds with information about the firmware version loaded into the transceivers.

The firmware version of all cables plugged into ports 1…40 of a switch with lid #nn can alternatively be checked with the mlxlink command:

```
# for i in {1..40}; do echo $i; mlxlink -d lid-nn -p $i -m | grep 'Part\|FW'; done
```

Checking successful burning and operation - Example:

It is essential to check that the links come up AFTER the cable FW is updated and reactivated. This can be done as follows:

```
# for i in {36..40}; do echo $i; mlxlink -d lid-nn -p $i -m | grep 'Part\|FW\|State'; done
```

The 'State' parameter was added to the query. The response has the following format (example):

```
# 36

State                              : Active

Vendor Part Number                 : MFS1S00-H010

FW Version                         : 38.100.59

37

State                              : Active

Vendor Part Number                 : MFS1S00-H010

FW Version                         : 38.100.59
```

```
38
State                            : Active
Vendor Part Number               : MFS1S00-H010
FW Version                       : 38.100.59
39
State                            : Active
Vendor Part Number               : MFS1S00-H010
FW Version                       : 38.100.59
40
State                            : Active
Vendor Part Number               : MFS1S00-H010
FW Version                       : 38.100.59
```

Querying the Firmware Image

To query the FW image on a device, use the following command line:

flint -d <device> q

To query the FW image in a file, use the following command line:

flint -i <image file> q

where:

device	Device on which the query is run.
image file	Image file on which the query is run.

Examples:

- Query the FW on the device.

  # flint -d /dev/mst/mt4133_pciconf0 query

- Query the FW image file.

  # flint -i 25408-2_42_5000-MCX354A-FCB_A2.bin query

- Security Attributes field in Query output:

  This field lists the security attributes of the device's firmware, where:

  - Secure-fw: This attribute indicates that this binary/device supports secure-firmware-updates. It means that only officially signed binaries can be loaded to the device from the host, and that the current binary is signed.
  - Signed-fw: This attribute indicates that that this binary is signed and that the device can verify digital signatures of new updates. However, unlike, secure-fw, there might still be methods to upload unsigned binaries to the device from the host.
  - debug: This attribute indicate that this binary is (or this device runs) a debug-version. Debug versions are custom made for specific data-centers or labs, and can only be installed after a corresponding debug-fw token is pushed to the device. The debug-fw-token, which is digitally signed, includes a list of the target devices MAC addresses.
  - dev: This attribute indicates that the firmware is signed with development (test) key.

- Default Update Method" field in Query Full output:{This field reflect the method which flint will use in order to update the device. The user can enforce a different method using the –no_fw_ctrl or the –ocr flags.The default methods are:

  - Legacy: flint will use the low level flash access registers.
  - fw_ctrl: flint will operate the 'firmware component update' state machine.

- Secure-boot attributes

- Secure-boot : This attribute indicates if the device supports secure-boot
- Life-cycle : This attribute indicates the current status of secure-boot
- Security-version:
    - For query on image: This attribute indicates the security-version of the image.
    - For query on device:
        - “EFUSE security version”: Indicates the security version of the device
        - “Image security version”: Indicates the security version of the image on the flash
        - Programming method: Indicates when the boot will program the “EFUSE security version” to be aligned with the “image security version”.

Verifying the Firmware Image

To verify the FW image on the Flash, use the following command line:

```
# flint -d <device> verify
```

To verify the FW image in a file, use the following command line:

```
# flint -i <image file> v
```

where:

device	Flash device to verify.
image file	Image file to verify.

Examples:

```
# flint -d /dev/mst/mt4115_pciconf0 v
```

```
# flint -i ./image_file.bin verify
```

Comparing the Binary Image

Binary comparison of the firmware image enables the user to verify that a given firmware image contains the image that matches the given device.

> **Note**
>
> Since ConnectX-4/ConnectX-4 Lx devices have iTOC (image specific) and dTOC (device specific) sections at the beginning of the device flash, and the MFA2 archive does not have the dTOC information by its definition, the binary comparison will ignore the device specific sections on the device.
>
> ```
> #flint -d <device> -i <fw image> --silent (optional) bc (or binary_compare)
> ```

The Verify Command on Encrypted Flash/Image

> **Note**
>
> The `verify` command on encrypted flash/image is applicable on adapter cards starting from ConnectX-7.

The flint tool supports the `verify` command on encrypted flash/image, as follows:

- When both the device and the image are given the verify command: verifying encrypted flash, the flint tool will execute binary-compare between the flash and the given image (the image is expected to be the one burnt on the device). In case of a device in recovery mode, the verify action is applicable before transcoding.
- When an encrypted device/image is given the verify command: only DTOC CRCs will be verified. In case a device is given, the verify is applicable in recovery mode only.

Performing Checksum Calculation on Image/Device

The flint utility allows performing an MD5 checksum on the non-persistent sections of the firmware image. For example: the sections that are changed when performing a firmware upgrade.

To perform a checksum on the flash, run the following command line:

flint -d <mst device> checksumTo perform a checksum on a firmware image, run the following command line:

flint -i <image file> checksumwhere:

device	Flash device to verify.
image file	Image file to verify.

Examples:

flint -i fw-ConnectX4Lx.bin checksum -I- Calculating Checksum ... Checksum: 68ddae6bfe42f87f09084f3f468a35c6

flint -d /dev/mst/mt4117_pciconf0 cs

-I- Calculating Checksum ...

Checksum: 68ddae6bfe42f87f09084f3f468a35c6

Managing an Expansion ROM Image

To burn an Expansion ROM image, run the following command:

flint -d <mst device> brom <image name>.mrom

The “brom” command installs the ROM image on the flash device or replaces an already existing one.

Example:

flint -d /dev/mst/mt4099_pci_cr0 brom example.mrom

Current ROM info on flash: N/A

New ROM info: type=PXE version=3.5.305 cpu=AMD64

Burning ROM image - OK

Restoring signature - OK

#

To read an expansion ROM image to a file, run the following command:

flint -d <mst device> rrom <image name>.rom

Example:

flint -d /dev/mst/mt4099_pci_cr0 rromexample.mrom

flint -d /dev/mst/mt4099_pci_cr0 q

Image type: FS2

FW Version: 2.42.5000

FW Release Date: 4.5.2017

Rom Info: type=PXE version=3.5.305 cpu=AMD64

Device ID: 4099

Description: Node   Port1   Port2   Sys image

GUIDs: f45214030001b8a0 f45214030001b8a1 f45214030001b8a2 f45214030001b8a3

MACs: f4521401b8a1   f4521401b8a2

VSD:

PSID: MT_1090120019

#

To remove the expansion ROM, run the following command:

flint -d <mst device> drom

Examples:

flint -d /dev/mst/mt4099_pci_cr0 drom

Removing ROM image - OK

Restoring signature - OK

Setting GUIDs and MACs

To set GUIDs/MACs/UID for the given device, use the ‘sg’ (set guids) command with the -guid(s), -uid and/or -mac(s) flags.

4th Generation (Group I) Devices

On 4th generation/Group I devices, the “sg” command can operate on both the image file and the image on the flash. When running the “sg” command on an image on the flash, if the GUIDs/MACs/UIDs in the image are non-blank, the flint will re-burn the current image using the given GUIDs/MACs/UIDs.

1. Change the GUIDs/MACs on a device:

   # flint -d /dev/mst/mt4099_pci_cr0 q

   -W- Running quick query - Skipping full image integrity checks.

   Image type: FS2

   FW Version: 2.42.5000

   FW Release Date: 4.5.2017

   Device ID: 4099

   Description: Node   Port1   Port2   Sys image

   GUIDs: f45214030001b8a0 f45214030001b8a1 f45214030001b8a2 f45214030001b8a3

   MACs: f4521401b8a1   f4521401b8a2

   VSD:

   PSID: MT_1090120019

   # flint -d /dev/mst/mt4099_pci_cr0 -guid 0x452140300abadaba -mac 0x300abadaba sg

```
-W- GUIDs are already set, re-burining image with the new GUIDs ...

You are about to change the Guids/Macs/Uids on the device:

                 New Values        Current Values

  Node GUID:       452140300abadaba   f45214030001b8a0

  Port1 GUID:      452140300abadabb   f45214030001b8a1

  Port2 GUID:      452140300abadabc   f45214030001b8a2

  Sys.Image GUID:  452140300abadabd   f45214030001b8a3

  Port1 MAC:           00300abadaba       f4521401b8a1

  Port2 MAC:           00300abadabb       f4521401b8a2

Do you want to continue ? (y/n) [n] : y

Burning FS2 FW image without signatures - OK

Restoring signature - OK

# flint -d /dev/mst/mt4099_pci_cr0 q

Image type:     FS2

FW Version:     2.31.5050

FW Release Date: 4.5.2014

Device ID:      4099

Description:    Node            Port1           Port2           Sys image

GUIDs:          452140300abadaba 452140300abadabb 452140300abadabc 452140300abadabd

MACs:                            00300abadaba     00300abadabb

VSD:

PSID:           MT_1090120019
```

2. Change the GUIDs/MACs on an image file:

```
# flint -i /tmp/image.bin q
```

```
Image type:      fs2
FW Version:      2.31.5050
FW Release Date: 4.5.2014
Device ID:       4099
Description:     Node            Port1           Port2           Sys image
GUIDs:           f45214030001b8a0 f45214030001b8a1 f45214030001b8a2 f45214030001b8a3
MACs:                             00300abadaba     00300abadabb
VSD:
PSID:            MT_1090120019
# flint -i /tmp/image.bin -guid 0002c9000abcdef0 -mac 02c90abcdef0 sg
You are about to change the Guids/Macs/Uids on the device:
                 New Values        Current Values
   Node GUID:      0002c9000abcdef0   f45214030001b8a0
   Port1 GUID:     0002c9000abcdef1   f45214030001b8a1
   Port2 GUID:     0002c9000abcdef2   f45214030001b8a2
   Sys.Image GUID: 0002c9000abcdef3   f45214030001b8a3
   Port1 MAC:         02c90abcdef0       00300abadaba
   Port2 MAC:         02c90abcdef1       00300abadabb
Do you want to continue ? (y/n) [n] : y
Restoring signature - OK# flint -i /tmp/image.bin q
Image type:      FS2
FW Version:      2.31.5050
FW Release Date: 4.5.2014
Device ID:       4099
```

Description: Node Port1 Port2 Sys image

GUIDs: 0002c9000abcdef0 0002c9000abcdef1 0002c9000abcdef2 0002c9000abcdef3

MACs: 02c90abcdef0 02c90abcdef1

VSD:

PSID: MT_1090120019

5th Generation (Group II) Devices

On 5th Generation (Group II) devices, the “sg” command can operate on both the image file and the image on the flash. When running the “sg” command on an image on the flash, -uid flag must be specified. For ConnectX-4, -guid/-mac flags can be specified. By default, 8 GUIDs will be assigned for each port starting from base, base+1 up until base+7 for port 1 and base+8 up until base+15 for port 2.

To change the step size and the number of GUIDs per port, specify guids_num=<num> step_size=<size> to the sg command.

1. Change GUIDs for device:

    # flint -d /dev/mst/mt4113_pciconf0 q

    Image type: FS3

    FW Version: 10.10.3000

    FW Release Date: 29.4.2014

    Description: UID GuidsNumber Step

    Base GUID1: 0002c903002ef500 8 1

    Base GUID2: 0002c903002ef508 8 1

    Base MAC1: 0002c92ef500 8 1

    Base MAC2: 0002c92ef508 8 1

    Image VSD:

    Device VSD: VSD

    PSID: MT_1240110019

```
# flint -d /dev/mst/mt4113_pciconf0 -uid 0002c123456abcd -ocr sg

-W- Firmware flash cache access is enabled. Running in this mode may cause the firmware to hang.

Updating GUID section - OK

Updating ITOC section - OK

Restoring signature   - OK

# flint -d /dev/mst/mt4113_pciconf0 q

Image type:      FS3

FW Version:      10.10.3000

FW Release Date: 29.4.2014

Description:     UID              GuidsNumber   Step

Base GUID1:      00002c123456abcd  8            1

Orig Base GUID1: 0002c903002ef500  8            1

Base GUID2:      00002c123456abd5  8            1

Orig Base GUID2: 0002c903002ef508  8            1

Base MAC1:       00002c56abcd      8            1

Orig Base MAC1:  0002c92ef500      8            1

Base MAC2:       00002c56abd5      8            1

Orig Base MAC2:  0002c92ef508      8            1

Image VSD:

Device VSD:      VSD

PSID:           MT_1240110019
```

Note

Orig Base GUID/MAC refers to the GUIDs/MACs located in the MFG(manufacture guids) section of the flash/image.

2. Change GUIDS for device (specifying guids_num and step_size):

```
# flint -d /dev/mst/mt4113_pciconf0 q

Image type:      FS3

FW Version:      10.10.3000

FW Release Date: 29.4.2014

Description:     UID              GuidsNumber  Step

Base GUID1:      0002c903002ef500  8           1

Base GUID2:      0002c903002ef508  8           1

Base MAC1:       0002c92ef500      8           1

Base MAC2:       0002c92ef508      8           1

Image VSD:

Device VSD:      VSD

PSID:            MT_1240110019

# flint -d /dev/mst/mt4113_pciconf0 -uid 0000000000000001 -ocr sg guids_num=2 step_size=1

-W- Firmware flash cache access is enabled. Running in this mode may cause the firmware to hang.

Updating GUID section - OK

Updating ITOC section - OK

Restoring signature   - OK

# flint -d /dev/mst/mt4113_pciconf0 q

Image type:      FS3

FW Version:      10.10.3000
```

```
FW Release Date: 29.4.2014
Description:     UID              GuidsNumber  Step
Base GUID1:      0000000000000001  2           1
Orig Base GUID1: 0002c903002ef500  8           1
Base GUID2:      0000000000000003  2           1
Orig Base GUID2: 0002c903002ef508  8           1
Base MAC1:       000000000001      2           1
Orig Base MAC1:  0002c92ef500      8           1
Base MAC2:       000000000003      2           1
Orig Base MAC2:  0002c92ef508      8           1
Image VSD:
Device VSD:      VSD
PSID:            MT_1240110019
```

3. Change GUIDs for image:

```
# flint -i /tmp/connect-ib.bin q
Image type:      FS3
FW Version:      10.10.3000
FW Release Date: 29.4.2014
Description:     UID              GuidsNumber  Step
Base GUID1:      0002c903002ef500  8           1
Base GUID2:      0002c903002ef508  8           1
Base MAC1:       0002c92ef500      8           1
Base MAC2:       0002c92ef508      8           1
Image VSD:
```

```
Device VSD:      VSD
PSID:            MT_1240110019
# flint -i /tmp/connect-ib.bin -uid 000123456abcd sg
Updating GUID section - OK
Updating ITOC section - OK
Restoring signature   - OK
# flint -i /tmp/connect-ib.bin q
Image type:      FS3
FW Version:      10.10.3000
FW Release Date: 29.4.2014
Description:     UID              GuidsNumber  Step
Base GUID1:      000000123456abcd  8           1
Orig Base GUID1: 0002c903002ef500  8           1
Base GUID2:      000000123456abd5  8           1
Orig Base GUID2: 0002c903002ef508  8           1
Base MAC1:       00000056abcd      8           1
Orig Base MAC1:  0002c92ef500      8           1
Base MAC2:       00000056abd5      8           1
Orig Base MAC2:  0002c92ef508      8           1
Image VSD:
Device VSD:      VSD
PSID:            MT_1240110019
```

4. Change GUIDs and MACs for the ConnectX-4 device:

```
# flint -d /dev/mst/mt4115_pciconf0 -guid e41d2d0300570fc0 -mac 0000e41d2d570fc0 -ocr sg
```

-W- Firmware flash cache access is enabled. Running in this mode may cause the firmware to hang.

Updating GUID section - OK

Updating ITOC section - OK

Restoring signature   - OK

flint -d /dev/mst/mt4115_pciconf0 q

Image type: FS3

FW Version: 12.0100.5630

FW Release Date: 23.3.2015

Description: UID   GuidsNumber

Base GUID: e41d2d0300570fc0   4

Base MAC: e41d2d570fc0   4

Image VSD:

Device VSD:

PSID: MT_2190110032

add note:

GUIDs and MACs can be changed separately on ConnectX4

Preparing a Binary Firmware Image for Pre-assembly Burning

In some cases, OEMs may prefer to pre-burn the flash before it is assembled on board.To generate an image for pre-burning for 4th generation (Group I) devices, use the mlxburn "-striped_image" flag. The "striped image" file layout is identical to the image layout on the flash, hence making it suitable for burning verbatim.When pre-burning, the GUIDs/MACs inside the image should be unique per device. The following are two methods to pre-burn an image. You can choose the best method suitable for your needs.

Method 1: Pre-burn an Image with Blank GUIDs/MACs

In this method, the image is generated with blank GUIDs and CRCs. The GUIDs are set after the device is assembled using the flint "sg" command. To set GUIDs take less than 1 second when running on an image with blank GUIDs (through a PCI device).

> **Note**
>
> A device that is burnt with blank GUIDs/MACs will not boot as a functional network device as long as the GUIDs/MACs are not set.

To pre-burn an image with blank GUIDs/MACs:

1. Generate a striped image with blank GUIDs.

```
# mlxburn -fw ./fw-ConnectX3-rel.mlx -./MCX354A-FCB_A2-A5.ini
-wrimage./fw-ConnectX3-rel.bin -striped_image -blank_guids
-I- Generating image ...
-I- Image generation completed successfully.
```

2. Burn the image to a flash using an external burner.
3. (Optional) After assembly, query the image on flash to verify there are no GUIDs on the device.

```
# flint -d /dev/mst/mt4099_pci_cr0 q
Image type:      FS2
FW Version:      2.31.5050
FW Release Date: 4.5.2014
Device ID:       4099
Description: Node             Port1            Port2
Sys image
GUIDs:       ffffffffffffffff ffffffffffffffff
ffffffffffffffff ffffffffffffffff
```

```
MACs:                         ffffffffffffffff
ffffffffffffffff
VSD:          n/a
PSID:         MT_1090120019


-W- GUIDs/MACs values and their CRC are not set.
```

4. Set the correct GUIDs. Since the image is with blank GUIDs, this operation takes less than 1 second.

```
# flint -d /dev/mst/mt4099_pci_cr0 -guid 0x0002c9030abcdef0 -mac
0x0002c9bcdef1 sg
```

5. Query the image on flash to verify that the GUIDs are set correctly.

```
sg# flint -d /dev/mst/mt4099_pci_cr0 q
Image type:       FS2
FW Version:       2.31.5050
FW Release Date: 4.5.2014
Device ID:        4099
Description:      Node             Port1            Port2
Sys image
GUIDs:            0002c9030abcdef0 0002c9030abcdef1
0002c9030abcdef2 0002c9030abcdef3
MACs:                                  0002c9bcdef1
0002c9bcdef2
VSD:              n/a
PSID:             MT_1090120019
```

Method 2: Pre-burn an Image with Specific GUIDs/MACs for Each Device

In this method, a “base” image is generated with arbitrary default GUIDs and then updated with the correct GUIDs for each device.

To pre-burn an image with specific GUIDs/MACs for each device:

1. Generate the base image with arbitrary default GUIDs.

```
# mlxburn -fw./fw-ConnectX3-rel.mlx -c ./MCX354A-FCB_A2-
A5.ini -wrimage ./fw-ConnectX3-rel.bin -striped_image
```

2. Per device, set the device specific GUIDs in the image.

```
flint -i ./fw-ConnectX3-rel.bin -guid 0x0002c9030abcdef0 -mac
0x0002c9bcdef1 -striped_image sg
```

3. (Optional) After assembly, query the image on flash to verify there are no GUIDs on the device.

```
sg# flint -i ./fw-ConnectX3-rel.bin -striped_image q
Image type:      FS2
FW Version:      2.31.5050
FW Release Date: 4.5.2014
Device ID:       4099
Description:     Node             Port1            Port2
Sys image
GUIDs:           0002c9030abcdef0 0002c9030abcdef1
0002c9030abcdef2 0002c9030abcdef3
MACs:                                 0002c9bcdef1
0002c9bcdef2
VSD:             n/a
```

```
PSID:            MT_1090120019
```

Now the fw-ConnectX3-rel.bin image can be pre-burned to the flash. After the assembly, the device would be fully functional.

Setting the VSD

To set the vsd for the given image/device (4th generation/Group I), use the sv command with -vsd flag.

Example:

```
# flint -d /dev/mst/mt4099_pci_cr0 -vsd "MELLANOX" sv
Setting the VSD     - OK
Restoring signature - OK

# flint -d /dev/mst/mt4099_pci_cr0 q
Image type:       FS2
FW Version:       2.31.5050
FW Release Date:  4.5.2014
Device ID:        4099
Description:      Node             Port1            Port2
Sys image
GUIDs:            f45214030001b8a0 f45214030001b8a1
f45214030001b8a2 f45214030001b8a3
MACs:                                  00300abadaba
00300abadabb
VSD:              MELLANOX
PSID:             MT_1090120019
```

Disabling/Enabling Access to the Hardware

The secure host feature enables ConnectX family devices to block access to its internal hardware registers. The hardware access in this mode is allowed only if a correct 64 bits key is provided.

> **Note**
>
> The secure host feature requires a DOCA_OFED driver installed on the machine.

5th Generation Devices

Secure Host can be enabled on 5th generation devices in one of the following manners:

1. Set the key:

   ```
   # flint -d /dev/mst/mt4115_pciconf0 set_key 18022018
   -I- Secure Host was enabled successfully on the device.
   ```

2. Disable HW access:

   ```
   # flint -d /dev/mst/mt4115_pciconf0 hw_access disable 18022018
   -I- Secure Host was enabled successfully on the device.
   ```

   If the key was not provided in the command line, an interactive shell will ask for it, and verifying it:

```
# flint -d /dev/mst/mt4115_pciconf0 set_key
Enter Key : ********
Verify Key : ********
-I- Secure Host was enabled successfully on the device.
```

Or

1. Disable the Secure Host (Enable HW access):

```
# flint -d /dev/mst/mt4115_pciconf0 hw_access enable 18022018
-I- The Secure Host was disabled successfully on the device.
And the same as previous, providing the key can be done in
interactive shell:
# flint -d /dev/mst/mt4115_pciconf0 hw_access enable
Enter Key : ********
-I- The Secure Host was disabled successfully on the device.
```

Flash Operations

Reading a Word from Flash

To read one dword from Flash memory, use the following command line:

```
# flint -d <device> rw addr
```

where:

device	The device the dword is read from.
addr	The address of the word to read.

Example:

```
# flint -d /dev/mst/mt4099_pci_cr0 rw 0x20
```

Writing a dword to Flash

To write one dword to Flash memory, use the following command line:

```
# flint -d <device> ww addr data
```

where:

device	The device the dword is written to.
addr	The address of the word to write.
data	The value of the word.

Example:

```
# flint -d /dev/mst/mt4099_pci_cr0 ww 0x10008 0x5a445a44
```

Writing a dword to Flash Without Sector Erase

To write one dword to Flash memory without sector erase , use the following command line:

```
# flint -d <device> wwne addr data
```

where:

device	The device the dword is written to..

addr	The address of the word to write.
data	The value of the word.

Example:

```
# flint -d /dev/mst/mt4099_pci_cr0 wwne 0x10008 0x5a445a44
```

Note that the result may be dependent on the Flash type. Usually, bitwise and between the specified word and the previous Flash contents will be written to the specified address.

Erasing a Sector

To erase a sector that contains a specified address, use the following command line:

```
# flint -d <device> e addr
```

where:

device	The device the dword is erased from.
addr	The address of a word in the sector that you want to erase.

Example:

```
# flint -d /dev/mst/mtusb-1 e 0x1000
```

Querying Flash Parameters

To query flash parameters use the following command line:

```
# flint -d <device> [-ocr] hw query
```

where:

device	The device to query.

Example:

```
# flint -d /dev/mst/mt4115_pciconf0 hw query
```

Firmware Timestamping for Multi-Host Environment

In a multi-host environment, every host can upgrade the NIC firmware. All hosts are treated equally and there is no designated host. Hence, there can be situations where one host will try to upgrade the firmware and another will try to downgrade; which may lead to two or more unnecessary server reboots. In order to avoid such situations, the administrator can add a timestamp to the firmware they want to upgrade to. Attempts to burn a firmware image with a timestamp value that is lower than the current firmware timestamp will fail.

Note

Firmware timestamping can be used on Connect-IB/ConnectX-4/ConnectX-4 Lx HCAs for controlling the firmware upgrade/downgrade flow.

Setting a Timestamp on Image

In order to set a timestamp on an image, run:

```
# flint -i ./fw-4115.bin timestamp set [UTC time]
```

Note

The user can either specify a combined date and time timestamp in UTC which conforms to ISO 8601, or let the tool use the machine's time for the timestamp.

Querying a Timestamp on Image

To view the timestamp that was set on the device, run:

```
# flint -d /dev/mst/mt4115_pciconf0 timestamp query
Current timestamp    : N/A. No valid timestamp found
Next timestamp       : 2015-12-21T10:58:23Z     12.15.0005
```

- "Current timestamp" represents the current running firmware timestamp. If "N/A" is visible, then the timestamp entry is invalid (example: first use of the feature or after resetting the timestamp).
- "Next timestamp" represents the next firmware that is allowed to be burnt on the HCA. Updating the "Next timestamp" requires an equal or newer timestamp to be provided.

Resetting a Timestamp on Device

To reset the timestamp that was set on the device, run:

```
# flint -d /dev/mst/mt4115_pciconf0 timestamp reset
```

Resetting the timestamp on device causes invalidation of both “Current timestamp” and “Next timestamp” fields.

Setting a Timestamp on Device

In case it is not possible to modify the firmware image, it is possible to set the timestamp directly on the device by specifying the timestamp and firmware version tied to it.

```
# flint -d /dev/mst/mt4115_pciconf0 timestamp set <UTC time> 
<Firmware version>
```

Querying a Timestamp on Device

To view the timestamp that was set on the device, run:

```
# flint -d /dev/mst/mt4115_pciconf0 timestamp query
Current timestamp : N/A. No valid timestamp found
Next timestamp : 2015-12-21T10:58:23Z 12.15.0005
```

- “Current timestamp” represents the current running firmware timestamp. If N/A is visible, then the timestamp entry is invalid (example: first use of the feature or after resetting the timestamp).
- “Next timestamp” represents the next firmware that is allowed to be burnt on the HCA. Updating the “Next timestamp” requires an equal or newer timestamp to be provided.

Resetting a Timestamp on Device

To reset the timestamp that were set on the device, run:

```
# flint -d /dev/mst/mt4115_pciconf0 timestamp reset
```

Resetting the timestamp on device causes invalidation of both “Current timestamp” and “Next timestamp” fields.

Important Notes

Please note the following:

- If a firmware image contains a timestamp, the burning tool will automatically attempt to set it on the device. If the operation succeeds, the firmware will be burnt.
- If a timestamp was only set on the device, the burning tool will prevent the burning of any firmware version different than the one set in the timestamp set operation.
- Lack of timestamp in both image and device will cause no checks to be performed.

flint/mlxburn Limitations

- When running flint/mlxburn via an MTUSB-1 device, a burn/query command may take up to 45 minutes to complete.
    - To accelerate the burn process add the flag -no_flash_verify to the command line which skips the flash verification step. This flag, however, does not verify if the image is burnt correctly.
- Burning an image to a ConnectX-3 adapter in Flash recovery mode may fail on some server types (that use PCIe spread spectrum). The tool may not be able to recognize the device’s PCI CONF0 or the image burn may not complete successfully.
    - To burn the device, use the MTUSB-1 connection.
- To load the newly burnt firmware image, a driver restart is required for ConnectX-3/ConnectX-3 Pro cards.
    - For fifth generation (Group II) devices, run the mlxfwreset tool or reboot the system.

Secure Host

Secure host is the general term for the capability of a device to protect itself and the subnet from malicious software through mechanisms such as blocking access of untrusted entities to the device configuration registers, directly (through pci_cr or pci_conf) and indirectly (through MADs).

When Secure Host is enabled, host is blocked from performing firmware updates and setting non-volatile configurations.

> **Note**
>
> The supported opcodes of secure_host register by flint tool is setting/unsettling the lock.

> **Warning**
>
> WARNING:
>
> - Once a hardware access key is set, the hardware can be accessed only after the correct key is provided.
> - If a key is lost, please refer to Key Loss Recovery.

> **Note**
>
> - The hardware access in this mode is allowed only if a correct 64 bits key is provided.
> - The secure host feature for ConnectX-3/ConnectX-3 Pro HCAs requires a MLNX_OFED driver installed on the machine.

Using Secure Host

Secue Host feature is supported for all NVIDIA® network adapters (listed in Group 1 and group 2). For group 1 network adapters, the user is required to generate and burn a firmware image that supports the feature (see "Generating/Burning a Firmware Supporting Secure Host" below).

For Group 2 network adapters, the feature is supported on firmware version 1x.22.1002 or newer.

Generating/Burning a Firmware Supporting Secure Host

> **Note**
>
> This is not applicable for ConnectX-4 devices and above. No burning of a specific firmware is needed.

1. Make sure you have INI and mlx files suitable for the device.
    1. Add cr_protection_en=true under [HCA] section in the INI file.
    2. Generate an image using mlxburn, for example run:

    ```
    # mlxburn -fw ./fw-4099-rel.mlx -conf ./edited_conf.ini -wrimage fw-4099.bin
    ```

2. Burn the image on the device using flint:

    ```
    # flint -d /dev/mst/mt4099_pci_cr0 -i fw-4099.secure.bin b
    ```

3. For changes to take effect, reboot is required.

Setting the Secure Host Key (for all NIC cards)

To set the key, run:

```
# flint -d <dev> set_key 22062011
Setting the HW Key - OK
Restoring signature - OK
```

Note

A driver restart is required to activate the new key.

Accessing the device after hardware access is disabled should show the following:

```
# flint -d <dev> q
E- Cannot open <dev>: HW access is disabled on the device.
E- Run "flint -d <dev> hw_access enable" in order to enable HW access.
```

Enabling Hardware Access

Back user should run the following:

For ConnectX-4 and above devices:

run:

```
# flint -d <device> hw_access enable
Enter Key: ********
```

For ConnectX-3 and ConnectX-3 Pro devices:

1. Make sure you have INI and MLX file suitable for the device.

1. Remove cr_protection_en=true from the INI (if present)

2. Generate the image using mlxburn, for example run:

```
# mlxburn -fw ./fw-4099-rel.mlx -conf ./unsecure_host.ini
-wrimage fw-4099.unsecure.bin
```

2. Burn the firmware on the device (make sure hardware access is enabled prior to burning):

```
# flint -d /dev/mst/mt4099_pci_cr0 -i fw-4099.unsecure.bin b
```

3. Execute a driver restart in order to load the unsecure firmware:

```
# service openibd restart
```

Key Loss Recovery

If a key is lost, there is no way to recover it using the tool. The only way to recover is to:

1. Connect the flash-not-present jumper on the card.

2. Reboot the machine.

3. Re-burn firmware(for Group 2 network adapters re-burn the firmware following the process in Burning a New Device.)

4. Remove the flash-not-present jumper.

5. Reboot the machine

6. Re-set the hardware access key

Secure Firmware Update

> **Note**
>
> Secure Firmware Update is supported only on ConnectX-4 onwards adapter cards.

A “Secure firmware update” is the ability of a device to verify digital signatures of new firmware binaries, in order to assure that only officially approved versions can be installed from the host, the network[1] or a Board Management Controller (BMC).

The firmware of devices with “secure firmware up date” functionality (secure FW), restricts access to specific commands and registers that can be used to modify the firmware binary image on the flash, as well as commands that can jeopardize security in general. Most notably, the commands and registers for random flash access are disabled.

Secure FW verifies new binaries before activating them, compared to legacy devices where this task is done by the update tool using direct flash access commands. In addition to signature verification, secure FW also checks that the binary is designated to the same device model, that the new firmware is also secured, and that the new FW version is not included in a forbidden versions blacklist. The firmware rejects binaries that do not match the verification criteria.

Secure FW utilizes the same ‘fail safe’ upgrade procedures, so events like power failure during update should not leave the device in an unstable state. The table below lists the impact of secure FW update on MFT tools.

Tool	Flow	Secure FW	With CS Token	Blocked Commands
flint / mlxburn	Burn FW	Working with controlled fw update	Working with controlled fw update	
	Query	Working with controlled fw update	Working with controlled fw update	
	Set GUIDs	Working with controlled fw update	Working with controlled fw update	

Tool	Flow	Secure FW	With CS Token	Blocked Commands
	Verify	Working partially (BOOT image)	Working partially (BOOT image)	
	Set DV INFO: SET MFG, SET VSD, VPD	Not supported in Secure FW	Not supported in Secure FW	MFBA
	ROM OPS: BROM, DROM	Not supported, BOOT image modification is not supported (MFBA)	Not supported, BOOT image modification is not supported (MFBA)	MFBA
	"-ocr" override cache replacement (Direct flash GW access)	Not supported in Secure FW	Not supported in Secure FW	Flash GW is blocked
	HW SET (Set flash parameters)	Flash GW is blocked	Flash GW is blocked	Flash GW is blocked
	"--no_fw_ctrl" (Legacy Flow)	Not supported in Secure FW	Not supported in Secure FW	MFBA
mlxfwmanager / mlxup	Burn FW	Working with controlled fw update	Working with controlled fw update	
mlxfwmanager	with --no_fw_ctrl	Not supported in Secure FW	Not supported in Secure FW	MFBA

Tool	Flow	Secure FW	With CS Token	Blocked Commands
mlxdump	fsdump	Blocked icmds	Working	gcif_get_ft_info, gcif_get_ft_lis gcif_get_fg, gcif_get_fg_list, gcif_get_fte, gcif_get_fte_list
	phyUc	Blocked icmds	working	gcif_phy_uc_get_array_prop_p: gcif_phy_uc_set_get_data, gcif_phy_uc_get_array_prop_El gcif_phy_uc_get_array_prop_H
	rxdump	CR-Space is locked & Blocked icmds	working	gcif_read_rx_slice_desc, gcif_read_rx_slice_packet
	sxdump	CR-Space is locked & Blocked icmds	working	gcif_read_wq_buf fer
wqdump	Dump QP contexts	Blocked icmds	working	gcif_read_context
	Dump WQs	Blocked icmds	working	gcif_read_host_m em, gcif_read_q_en- try, gcif_qp_get_pi_ci
	ICM	Blocked icmds	working	gcif_read_icm
	WRITE QP (Devmon)		working	gcif_write_context
mget_temp	hw_access	Read Only CR- Space	working	Read Only CR- Space
mcra	Read	working	working	working
	Write	Read Only CR- Space	working	Read Only CR- Space
mstdump	Read	working	working	working
mlxtrace / fwtrace	MEM & FIFO	Only fwtrace is supported and only in Linux	working	Read Only CR- Space

Tool	Flow	Secure FW	With CS Token	Blocked Commands
pckt_drop	uses write to CR-Space to work	Read Only CR- Space	working	Read Only CR- Space
mlxlink	working	working	working	working
mlxreg	working	working	working	working
mlxcables	working	working	working	working
mlxconfig	working	working	working	working
mlxfwreset	working	working	working	working
i2c/mlxi2c	Not relevant when not in livefish			
	With Force flag (ENV VAR)	Read Only CR- Space	working	Read Only CR- Space

Secure Firmware Implications on Burning Tools

When Secure Firmware is enabled, the flint output slightly changes due to the differences in the underlying NIC accessing methods. Some functionalities may be restricted according to the device security level.

flint query under secure mode:

```
# flint -d /dev/mst/mt4115_pciconf0 q
Image type:       FS3
FW Version:       12.19.2278
FW Release Date: 7.6.2017
Description:      UID                 GuidsNumber
Base GUID:        7cfe90030029205e    4
Base MAC:         00007cfe9029205e    4
Image VSD:
Device VSD:
PSID:             MT_2190110032
```

```
Security Attributes: secure-fw, dev
```

> **Note**
>
> Unavailable information is reported as N/A.

Burning/Querying a Component

Burning a Component Firmware Image

Clock Synchronizer Images

The flint utility enables the user to burn the Clock Synchronizer firmware from a binary image.

```
# flint --device <mst device> --image <clock synchronizer image>
burn
```

Where:

-d|--device	mst device
-i|--image	Specified component firmwarew image file to use.

Querying the Component Firmware Image

Clock Synchronizer Images

To query the Clock Synchronizer image on a device, use the following command line:

```
# flint --device <mst device> --component_type sync_clock
query_components
```

To query the Clock Synchronizer image in a file, use the following command line:

```
# flint --image <image file> query
```

mlxburn – Firmware Image Generator and Burner

mlxburn is a tool for firmware (FW) image burning to the Flash/EEPROM attached to an NVIDIA device. It can also query for firmware attributes (e.g., firmware version, GUIDs, etc.) and VPD info of adapter cards and switch systems.

mlxburn Synopsis

```
#mlxburn [-h][-v] <-dev mst-device> <-img_dir img_direcory> [-
force][-fwver][-vpd][-vpd_rw][-vpd_prog_rw [-query]
```

where:

-dev <mst-dev>	Burn the image using the given mst device
-fwver	When a device is given: Display current loaded firmware version. When a FW file is given (-fw flag): Display the file FW version.Note: The “-fwver” flag is not supported in Connect-IB devices.
-h	Display a short help text
-image <fw-image-file>	Do not generate image. Use the given fw image instead

-striped_image	When specified, generated image will be in striped format, and will indicate that the image is in striped format when queried.
-query	Query the HCA/Switch device for firmware details, e.g. Firmware Version, GUIDs, etc. In addition to the above flags, Mlxburn can also accept the following flags/options, which are passed to the underlying burning tool: -banks -use_image_ps -skip_is-mac(s) -guid(s) -sysguid-vsd -ndesc -bsn-pe_i2c -se_i2c -is3_i2c-no -uid(s)-log -blank_guids -flash_params-allow_psid_change -no_flash_verify-use_image_rom -override_cache_replacement-use_image_guids See the flint tool documentation for HCA/4th gen switches/Bridge burning options.
-v	Print version info and exit
-V <INFORM|WARNING|DEBUG>	Set verbosity level. Default is WARNING
-vpd [1,2]	Display the read only section of the PCI VPD (Vital Product Data) of the given device
-vpd_rw [1,2]	**(on Linux only)**: Display also the read/write section of the PCI VPD of the given device.

Note 1. The VPD query may not be enabled on certain board types. Also, VPD operations are available only for devices with a PCI interface.

Note 2. Running multiple VPD access commands in parallel on the same device, by mlxburn or any other VPD access tool, may cause the commands to fail. VPD access commands should be run one at a time.

Connect-IB, Switch-IB, Switch-IB 2, NVIDIA Spectrum, ConnectX-4, and ConnectX-4 Lx Initial Burning Options

The following options are relevant when generating an image for initial burning. The image contains the VPD and the GUIDs that are in a read-only area on flash.

```
[ -vpd_r_file <vpd_r_file>] [ -base_guid <GUID>]
```

where:

-vpd_r_file <vpd_r_file>	Embed the given VPD Read-Only section in the generated image. The vpd_r_file should contain the vpd read only section and the first dword of the vpd writeable section. The file is in binary format, and its size must be a multiple of 4 bytes. Please refer to PCI base spec for VPD structure info.
-base_guid <GUID>	Set the given GUID as the image base GUID. The base GUID is used to derive GUIDs and MACs for the HCA ports. It is assumed that 16 GUIDs (base_guid to base_guid + 15) are reserved for the card. Note: On ConnectX-4/ConnectX-4 Lx/ConnectX-5/ConnectX-5 Ex, only GUIDs will be derived according to the HCA configuration.
-base_mac <MAC>	Set the given MAC as the image base MAC. The base MAC is used to derive MACs for the HCA ports according to the device configuration (ConnectX-4/ConnectX-4 Lx/ConnectX-5/ConnectX-5 Ex).
-vsd <string>	Write this string, of up to 208 characters, to VSD section.

Additional mlxburn Options

The following is a list of additional options. Please see mlxfwmanager – Firmware Update and Query Tool for the HCA options.

```
-banks  -use_image_ps  -skip_is  -mac(s)  -guid(s) -sysguid  -
ndesc  -bsn  -use_image_guids  -pe_i2c  -se_i2c  -is3_i2c  -no  -
qq  -uid(s)  -log  -blank_guids  -flash_params  -
allow_psid_change  -no_flash_verify  -use_image_rom
-override_cache_replacement  -ocr  -ignore_dev_data  -use_dev_rom
-no_fw_ctrl
```

Note

The arguments of the -guids and -macs flags must be provided within quotation marks; for example, mlxburn -macs “0002c900001 0002c900002”.

Examples of mlxburn Usage

Query for firmware attributes (e.g., firmware version, GUIDs, etc.):

```
# mlxburn  -d /dev/mst/mt4129_pciconf0  query
-I- Image type:            FS4
-I- FW Version:            28.45.0376
-I- FW Release Date:       15.4.2025
-I- Product Version:       28.45.0376
-I- Rom Info:              type=UEFI version=14.38.14
cpu=AMD64,AARCH64
-I-                        type=PXE version=3.7.500 cpu=AMD64
-I- Description:           UID                GuidsNumber
-I- Base GUID:             N/A                    8
-I- Base MAC:              N/A                    8
-I- Image VSD:             N/A
-I- Device VSD:            N/A
-I- PSID:                  MT_0000000851
-I- Security Attributes:   secure-fw
```

VPD read:

```
# mlxburn -d /dev/mst/mt4123_pciconf0 -vpd_rw
```

```
  VPD-KEYWORD    DESCRIPTION            VALUE
  -----------    -----------            -----
Read Only Section:

  PN             Part Number            MCX653106A-HDAT
  EC             Revision               A1
  V2             N/A                    MCX653106A-HDAT
  SN             Serial Number          MT1836X04719
  V3             N/A                    0
  VA             N/A
MLX:MN=MLNX:CSKU=V2:UUID=V3:PCI=V0:MODL=CX653106A
  V0             Misc Info              PCIeGen4 x16
  VU             N/A                    MT1836X04719MLNXS0D0F0
  RV             Checksum Complement    0xc0
  IDTAG          Board Id               CX653106A - ConnectX-6
QSFP56

Read Write Section:
```

```
# mlxburn vpd -d /dev/mst/mt4123_pciconf0

  VPD-KEYWORD    DESCRIPTION            VALUE
  -----------    -----------            -----
Read Only Section:

  PN             Part Number            MCX653106A-HDAT
  EC             Revision               A1
  V2             N/A                    MCX653106A-HDAT
  SN             Serial Number          MT1836X04719
  V3             N/A                    0
  VA             N/A
MLX:MN=MLNX:CSKU=V2:UUID=V3:PCI=V0:MODL=CX653106A
  V0             Misc Info              PCIeGen4 x16
```

```
  VU              N/A                     MT1836X04719MLNXS0D0F0
  RV              Checksum Complement     0xc0
  IDTAG           Board Id                CX653106A - ConnectX-6
QSFP56
```

Burning bin file using mlxburn:

```
# mlxburn -d /dev/mst/mt4131_pciconf0 -i cx8.bin -y
```

Detecting and burning the suitable bin file from a dir of several bin files:

```
# mlxburn -d /dev/mst/mt4131_pciconf0 -img_dir /tmp/s -y
```

To Query fw version of the given device:

```
# mlxburn -d /dev/mst/mt4115_pciconf0 -fwver
-I- FW Version: 12.28.2006
```

Exit Return Values

The following exit values are returned:

- 0 - successful completion
- >0 - an error occurred

mlxfwreset – Loading Firmware on 5th Generation Devices Tool

- The mlxfwreset tool enables the user to load updated firmware on a NIC/switch without having to reboot the machine. The mlxfwreset tool supports 5th Generation (Group II) HCAs and allows a smooth firmware upgrade.

> **Note**
>
> This feature is supported if either reset type 3 (ARM RESET), which is the default starting from this version, or 4 (ARM SHUT DOWN), and reset level 1 (IMMEDIATE RESET) are supported.

Tool Requirements

- Access to device through PCI configuration cycles
- Supported OSs: FreeBSD, Linux, Windows

Query Command

```
mlxfwreset -d <device> query
```

Reset Command

```
mlxfwreset -d <device> reset-[y] [--level <0,3,4>] [--type
<0..2>] [--sync <0,1,2>] [-s] [-m] [--method <0,1>
```

mlxfwreset Synopsis

Where:

q|query	Query supported reset level/type/sync.	N/A for switch devices.
r|reset	Execute reset.	-
reset_fsm_register	Reset the multi-host synchronization register.	-
-d|--device <device>	Device to work with.	-
-l|--level <0,1,3,4>	Run reset with the specified reset-level.	N/A for switch devices.
-t|--type <0..4>	Run reset with the specified reset-type.	N/A for switch devices.
--sync <0,1,2>	Run reset with the specified reset-sync.	N/A for switch devices.
--method <0,1>	Run reset with the specified request method (relevant only for reset-level 3)	-
-y|--yes	Answer “yes” on prompt.	-
-m|--mst_flags MST_FLAGS	Provide mst flags to be used when invoking mst restart step. For example: --mst_flags="–with_fpga".	◦ This option is supported in Linux OSes only. ◦ N/A for switch devices.
-s|--skip_driver	Skip driver start/stop stage (driver must be stopped manually).	N/A for switch devices.
-v|--version	Print tool version.	-
-h|--help	Show help message and exit.	-

Reset Levels and Types

Reset levels and types depend on the extent of the changes introduced when updating the device's firmware. The tool will display the supported reset levels and types that will ensure the loading of the new firmware. Those reset levels and types are:

- Reset-levels:

        - 0: Driver, PCI link, network link will remain up ("live-Patch")
        - 1: Only ARM side will not remain up ("Immediate reset")
        - 3: Driver restart and PCI reset
        - 4: Warm Reboot
    - Reset-types (relevant only for reset-levels 1, 3, 4):
        - 0: Full chip reset
        - 1: Phy-less reset ("port-alive" - network link will remain up) – Not Suppported
        - 2: NIC only reset (for SoC devices)
        - 3: ARM only reset
        - 4: ARM OS shut down

> **Note**
>
> The exact reset-level and reset-types needed to load new the firmware may differ as they depend on the difference between the running firmware and the firmware we are upgrading to.

Reset Sync

- Reset-sync indicates who is responsible for the synchronization mechanism between the hosts on the Multi-Host setup (relevant only for reset-level 3):
    - 0: Tool is the owner
    - 1: Driver is the owner
    - 2: FW is the owner

Reset Method

- Reset request method (relevant only for reset-level 3):
    - 0: Link Disable method
    - 1: Hot reset (SBR)

PCIe Switch Reset via Hot Reset

- We provide a mechanism to reset a PCIe switch using a hot reset, eliminating the need to reboot the entire setup.

  This is achieved using sync 2 and method 1 arguments, which together define the hot reset flow.

  This feature currently supports ConnectX-7 and BlueField devices.

  To trigger the reset, users can either run the tool with: --sync 2 --method 1 arguments or simply run it without any parameters and the tool will apply the default values automatically.

- **Important:**

  Before executing the reset, **the user should stop any driver bound or associated with devices downstream to the PCIe Switch**. The tool may also prompt the user to unload specific drivers or terminate some processes that it is aware of, b**ut it is not necessarily aware of all the bounded drivers.**

  It is the user responsibility to ensure these steps are completed to guarantee a successful reset.

mlxfwreset for Switch Devices

Running mlxfwreset on a switch device is done in the same form as running mlxfwreset on a NIC. The only difference is that there are no level, types or sync parameters.

mlxfwreset for Multi-Host NICs

Running mlxfwreset on a Multi-Host setup enables you to choose one of the supported reset-sync. To check which reset-sync are supported on your device, run the query command prior to the reset command.

- When running reset with reset-sync "0" ("tool is the owner"), the tool must be ran simultaneously on all the hosts. Note that a time-out of 3 minutes is expected for all the hosts until they join the reset process.

- When running reset with reset-sync "1" ("driver is the owner"), the tool must be ran on a single host.

> **Note**
>
> reset-sync "1" ("driver is the owner") is supported only when the firmware and all the drivers on all the hosts support it.

mlxfwreset for SmartNICs (Bluefield)

After upgrading the FW , you can run mlxfwreset with level 3 and sync 1.

If you use Level 4 standalone (without shutting down the ARM), you need to execute mlxfwreset on both the integrated ARM and the host.

To update the FW using Level 4, the ARM must be shut down first.

> **Note**
>
> Depending on the reset-type, the integrated Arm might get reset. In case Arm is reset, the mlxfwreset on the host will wait for the Arm to complete the reboot process.

mlxfwreset after Changing Configurations using mlxconfig

Some configuration changes require PCI rescan by the user, in this case, mlxfwreset will print the following warning message:

`"-W- PCI rescan is required after device reset."`

Examples of mlxfwreset Usage

To query the default and supported options to reset a device, run:

```
# mlxfwreset -d /dev/mst/mt4113_pciconf0 query
```

Example:

```
Reset-levels:
0: Driver, PCI link, network link will remain up ("live-Patch")  -
Not Supported
1: Only ARM side will not remain up ("Immediate reset").        -
Not Supported
3: Driver restart and PCI reset
-Supported      (default)
4: Warm Reboot
-Supported

Reset-types (relevant only for reset-levels 1,3,4):
0: Full chip reset
-Supported      (default)
1: Phy-less reset ("port-alive" - network link will remain up)   -
Not Supported
2: NIC only reset (for SoC devices)
  -Not Supported
3: ARM only reset
-Not Supported
4: ARM OS shut down
-Not Supported

Reset-sync (relevant only for reset-level 3):
0: Tool is the owner
-Supported      (default)
1: Driver is the owner
-Supported
```

```
In the new mlxfwreset for BF2 and BF3, sync1 is the default.
Sync0 is not supported.
```

To reset the device in order to load the new firmware, run:

```
# mlxfwreset -d /dev/mst/mt4113_pciconf0 reset
```

Or:

```
mlxfwreset -d /dev/mst/mt53100_pciconf0 reset -y
```

Example

```
3: Driver restart and PCI reset
Continue with reset?[y/N] y
-I- Stopping Driver                 -Done
-I- Sending Reset Command To Fw     -Done
-I- Resetting PCI                   -Done
-I- Starting Driver                 -Done
-I- Restarting MST                  -Done
-I- FW was loaded successfully.
```

To reset a device with a specific reset level to load new firmware, run:

```
# mlxfwreset -d /dev/mst/mt4113_pciconf0 -l 4 reset
```

Example

```
Requested reset level for device, /dev/mst/mt4113_pciconf0:
4: Warm Reboot
Continue with reset?[y/N] y
-I- Sending reboot command to machine
```

mlxfwreset Limitations

The following are the limitations of mlxfwreset:

- Executing a reset-level or reset-type or reset-sync that is not supported (as shown in the query command) will yield an error
- When burning firmware with flint/mlxburn at the end of the burn the following message is displayed:

  -I- To load new FW run mlxfwreset or reboot machine

  If this message is not displayed, a reboot is required to load a new firmware
- On an old firmware, after a successful reset execution, attempting to query or reset again will yield an error as the load new firmware command was already sent to the firmware
- In case mlxfwreset exits with error after the “Stopping driver” step and before the “Starting driver” step, the driver will remain down. The user should start the driver manually in this case
- mlxfwreset for switch devices does not work over InfiniBand or remote connection

mlxphyburn – Burning Tool for Externally Managed PHY

Mlxphyburn tool allows the user to burn firmware of an externally managed PHY. The tool burns and verifies a pre-compiled binary PHY firmware image on the PHY’s flash. It is supported only on Linux.

Tool Requirements

- ConnectX-3/ConnectX-3 Pro with an externally managed PHY
- A device that has access to the PHY flash module
- MLNX_OFED driver (if installed) must be down
- Access to the device through the PCI interface (pciconf/pci_cr)
- Firmware version that supports access to an externally managed PHY
    - Version 2_33_5000 and above

mlxphyburn Synopsis

```
# mlxphyburn [-d <device>] [-i Phy_fw_image] b[urn]|q[uery]
```

where:

-d|--dev <device>	Device which has access to the PHY.
-i|--img <PHY_fw_image>	PHY firmware image.
-v|--version	Display version info.
-h|--help	Display help message.
b[urn]	Burn given PHY image on the device's PHY.
q[uery]	Query PHY FW on device.

Note

If no device is specified, mlxphyburn will attempt to burn the PHY firmware image on all mst devices on the machine.

Examples of mlxphyburn Usage

Burn Example

```
# mlxphyburn -d /dev/mst/mt4099_pciconf0 -i
Firmware_1.37.10_N32722.cld burn
-I- attempting to burn PHY Fw on device: /dev/mst/mt4099_pciconf0
-I- Burning...(Process might take a few minutes)
-I- Device burned and verified.
```

Query Example

```
# mlxphyburn -d /dev/mst/mt4099_pciconf0 q
-I- Querying device: /dev/mst/mt4099_pciconf0
Flash Type   : Atmel AT25DF041A
FW version   : 1.37
Image ID     : 1.37.10 InterfaceMasters N32722 Apr 14, 2014 12:21:00
Image ROM ID : 0
```

mlx_fpga – Burning and Debugging Tool for NVIDIA Devices with FPGA

> **Note**
>
> The mlx_fpga utility will be deprecated as of MFT v4.18.0.

mlx_fpga tool allows the user to burn and update a new FPGA image on NVIDIA Innova ™ adapter cards. For instructions on how to burn, please refer to Burning the FPGA's Flash Device using the mlx_fpga Burning Tool.

The tool also enables the user to read/write individual registers in the FPGA configuration space.

Tool Requirements

- An Innova IPsec 4 Lx EN / Innova Flex 4 Lx EN adapter card with an FPGA device
- For Innova IPsec 4 Lx EN: Load the mlx5_fpga_tools module
- For Innova Flex 4 Lx EN: Load the mlx_accel_tools module

  Note: The module is included in the Innova driver which is supplied for this product line only, and available through the Support.

- Start mst service with the fpga lookup flag (mst start --with_fpga)

mlx_fpga Synopsis

where:

-d|--device <device>	FPGA mst device interface
-v|--version	Display version info
-h|--help	Display help message
-f|--force	Non-interactive mode, answer yes to all questions
r |read <addr>	Read debug register in address
w |write <addr> <data>	Write data to debug register in address
b |burn <image file/s>	Burn the image on the flash
l |load	Load image from flash (--factory - load image from factory flash)
clear_semaphore	Unlock flash controller semaphore
reset	Reset flash controller (--gw) or FPGA (--fpga)
q |query	Query general FPGA information
set_fw_mgmt <disable|enable>	Disable/Enable FPGA management by the Firmware

Examples of mlx_fpga Usage

Adding FPGA mst Device Interface

For Innova IPsec 4 Lx EN: Load the mlx5_fpga_tools module.

```
# modprobe mlx5_fpga_tools
# mst start --with_fpga
# mst status
MST modules:
------------
MST PCI module is not loaded
MST PCI configuration module is not loaded
MST devices:
------------
No MST devices were found nor MST modules were loaded.
You may need to run 'mst start' to load MST modules.
FPGA devices:
-------------------
/dev/mst/mt4117_pciconf0_fpga_i2c
/dev/mst/mt4117_pciconf1_fpga_rdma1
```

Note: In the last line, it is recommended to use the RDMA device as it is faster. I2C is used for recovery purposes when RDMA is not functional.

For Innova Flex 4 Lx EN: Load the mlx_accel_tools module.

```
# modprobe mlx_accel_tools
# mst start --with_fpga
# mst status
MST modules:
------------
MST PCI module is not loaded
MST PCI configuration module is not loaded
MST devices:
------------
No MST devices were found nor MST modules were loaded.
You may need to run 'mst start' to load MST modules.
```

```
FPGA devices:
-------------------
/dev/mst/mt4117_pciconf0_fpga_i2c
/dev/mst/mt4117_pciconf1_fpga_rdma
```

Note: In the last line, it is recommended to use the RDMA device as it is faster. I2C is used for recovery purposes when RDMA is not functional.

For recovery only, where modules are broken or missing.

```
# mst start --with_fpga_fw_access
Starting MST (Mellanox Software Tools) driver set
Loading MST PCI module - Success
Loading MST PCI configuration module - Success
Create devices
Unloading MST PCI module (unused) - Success
# mst status
MST modules:
------------
MST PCI module is not loaded
MST PCI configuration module loaded
MST devices:
------------
/dev/mst/mt4117_pciconf0         - PCI configuration cycles access.
                                   domain:bus:dev.fn=0000:01:00.0 addr.reg=88 data.reg=92
                                   Chip revision is: 00
FPGA devices:
-------------------
/dev/mst/mt4117_pciconf0_fpga
```

Note

The mlx5_fpga_tools and the mlx_accel_tools modules must be down before trying to recover the FPGA mst Device Interface.

Burning the FPGA's Flash Device using the mlx_fpga Burning Tool

mlx_fpga tool burns a .bin file onto the FPGA flash device.

The Innova Flex .bin file must be generated according to the instructions listed in the *Innova Flex Adapter Card User Manual*.

Note

It is recommended to burn the FPGA device using an RDMA device as it faster and it shortens the burning time.

1. Burn the image.

```
# mlx_fpga -d <device> burn image.bin
```

2. Load the FPGA image from flash according to "Loading the Tool" below or power cycle the machine for change to take effect.

Loading the Tool

Load an FPGA image from user configurable flash:

```
# mlx_fpga -d <device> l/load <optional: load options>
```

where <optional: load options> is:

--factory	Load FPGA image from factory flash
--user	Load FPGA image from user flash [Default option]

Debugging the Tool

Reading One Debug Register:

```
# mlx_fpga -d <device> read 0x0
```

Writing One Debug Register:

```
# mlx_fpga -d <device> write 0x0 0x0
```

Updating the FPGA Image

In order to verify the new image burned to the FPGA, the user can use mlx_fpga tool and read the following registers:

Name	Address	Range	Default	RW	Description
image_version	0x900000	31:00:00	0x0	RO	Version of the image increased on every synthesis.
image_date	0x900004	31:00:00	0x0	RO	Image date of creation. The hex number is actually the decimal value, i.e. 0x12011995 means 12/01/ 1995 in DD/MM/YY:

Name	Address	Range	Default	RW	Description
					bits [31:24] = day of creation bits [23:16] = month of creation bits [15:0] = year of creation
image_time	0x900008	31:00:00	0x0	RO	Image time of creation. The hex number is actually the decimal value, i.e. 0x00015324 means 01:53:24 in HH:MM:SS: bits [23:16] = hour (00..23)bits [15:8] = minutes (00..59)bits [7:0] = seconds (00..59)

Note

Innova Flex users should refer to the Innova User Manual for more information.

cpldupdate – Tool for Programming On-Board CPLDs on NVIDIA Devices

cpldupdate tool allows the user to program on-board CPLDs on supporting NVIDIA products. The on-board CPLDs, as well as the core engine for programing them, are provided by Lattice Semiconductors.

cpldupdate accepts VME files as input and programs the appropriate CPLD on the device. The CPLD ID is embedded in the VME file. The user need not be aware of the board composition.

A VME file is the data file for use with the ispVM Embedded programming software. The file is essentially a binary version of an SVF file. SVF commands and data are stored in a binary format (with optional compression) for efficient storage and processing by embedded microprocessors. The ispVM Embedded software, provided as source code in C, interprets the VME data to manipulate the JTAG signals of connected target devices.

cpldupdate tool was provided by Lattice Semiconductors and was modified to fit NVIDIA needs.

For NVIDIA Quantum and Spectrum-2 switches, the CPLD update can be performed via GPIO instead using the firmware interface. By default, cpldupdate will use GPIO for Quantum and Spectrum-2 switches (if `"--dev"` option is specified), However, the user can use the `"--fw"` option to run cpldupdate via the firmware. Alternatively, the user can dismiss the `"--dev"` flag and use the `"--gpio"`.

For NVIDIA Spectrum-1 switches (SN2201) , the CPLD update must be performed via GPIO. The user must use the "--uncustomized" option in addition to the "--gpio" option in this case.

Tool Requirements

- CPLD bearing board

cpldupdate Synopsis

```
# cpldupdate [option] vme_file [vme_file]
```

where:

--dev <device>	<device> (e.g. lid-N, /dev/mst/mt4115_pciconf0).
--gpio	Update CPLD using GPIO.
--fw	Update CPLD using FW.
--idcode <num_of_bits>	Run IDCODE command and exits.
--print-progress	Print progress indication.
--uncustomized	Update CPLD via GPIO on NVIDIA Spectrum-1 (SN2201) switches.
--cpld_chain	Provides an option to select which chain to burn, possible chains are CPLD, FPGA and full chain.

Burn Example

```
# cpldupdate --dev /dev/mst/mt52000_pciconf0
./cpld000039_v0100.vme
Lattice Semiconductor Corp.
ispVME(tm) V12.2 Copyright 1998-2012.
Customized for Mellanox products.
Processing virtual machine file (./cpld000039_v0100.vme)......
+=======+
| PASS! |
+=======+
```

mlxprivhost - NIC Configuration by the Host Restriction Tool (Zero Trust Mode)

mlxprivhost enables the user to restrict the hosts from managing the device in case the BlueField DPU is in a Zero Trust environment, and the host cannot be considered “trusted”.

Note

mlxprivhost is supported in Linux only.

Note

mlxprivhost is not supported in ESXi 7.0.

Note

This utility is supported in BlueField devices only.

mlxprivhost Synopsis

```
mlxprivhost [OPTIONS] <command> [parameters…]
```

- Restrict configuration takes effect immediately, but **disabling/enabling RShim** requires power cycle
- A zero-trust (restricted) host will not be able to perform operations that can compromise the DPU, such as:
- Port ownership – the host cannot assign itself as port owner
- Hardware counters – the host does not have access to hardware counters
- Tracer functionality is blocked
- RShim interface is blocked
- FW flash is restricted
- For Multi-host systems, the tool is compatible with firmware versions starting from xx.31.10xx and later

where:

-h, --help	Shows this help message and exit
-v, --version	Shows program's version number and exit
-d <dev>, --device <dev>	Device to work with.
--disable_rshim	When TRUE, the host does not have an RSHIM function to access the embedded CPU registers (power cycle is required to apply changes)
--disable_tracer	When TRUE, the host will not be allowed to own the Tracer (requires FW reset to be applied)

--disable_counter_rd	When TRUE, the host will not be allowed to read Physical port counters (requires FW reset to be applied)
--disable_port_owner	When TRUE, the host will not be allowed to be Port Owner (requires FW reset to be applied)
r,restrict	Set all external hosts as zero-trust (restricted) except of the one that called the command
p,privilege	Set all external hosts privileged except the one that called the command
q,query	From external HOST: query the status of the host From Embedded ARM CPU: query the status of all external hosts.
-f, --full	Run with query command for high verbosity level - valid from embedded ARM CPU only.

Example of mlxprivhost:

- Enabling Zero-Trust host (Full Host Restriction - Embedded ARM CPU Only):

```
mlxprivhost –d /dev/mst/mt41682_pciconf0 r --disable_rshim --
disable_tracer --disable_counter_rd --disable_port_owner
```

- Disabling Zero-Trust host restriction (Embedded ARM CPU Only):

```
mlxprivhost –d /dev/mst/mt41682_pciconf0 p
```

- Query the status of the host\hosts (the full flag valid for embedded ARM CPU Only):

```
mlxprivhost -d /dev/mst/mt41682_pciconf0 q --full
Host configurations
-------------------
host index                        : 0
1                        2                        3
```

```
level                            : PRIVILEGED    PRIVILEGED
PRIVILEGED      PRIVILEGED

Port functions status:
-----------------------
disable_rshim           : FALSE                 FALSE
FALSE           FALSE
disable_tracer          : FALSE                 FALSE
FALSE           FALSE
disable_port_owner      : FALSE                 FALSE
FALSE           FALSE
disable_counter_rd      : FALSE                 FALSE
FALSE           FALSE
```

mlxtokengenerator - Token Creation Tool

The mlxtokengenerator tool allows the user to create Token XML files automatically. The Token XML file will be filled with all the required data. The generated Token XML file is ready to be used for sign and installation commands.

mlxtokengenerator Synopsis

```
# mlxtokengenerator [Options] <commands>
```

where:

-d|--dev <device>	Performs operation for a specified mst device
-t|--device_type <Switch/HCA/LinkX>	Mst device type
-k|--token_type <CS/DBG/CRCS/CRDT/RMCS/RMDT>	Token type

-o|--output_file <Path>	Path to output Token XML file
-f|--debug_fw <Path>	Path to debug fw file
-n| --nested_token	Get nested token for Retimer
-e|--nested_debug_fw	Path to debug fw file for Retimer nested token
-p|--tokens_dir <Path>	Path to a directory of tokens for aggregation
-v|--version	Displays version info
-h|--help	Displays help message
-s|--i2c_secondary <i2c_secondary>	I2C secondary address
--b|--blob	Path to challenge blob file (relevant for CRCS/CRDT token types)

Examples:

To create a CS token:

```
# mlxtokengenerator -d /dev/mst/mt4123_pciconf0 -k CS -t HCA -o
/tmp/cs_token.xml generate_token
```

To create a DBG token:

```
# mlxtokengenerator -d /dev/mst/mt4123_pciconf0 -k DBG -t HCA -f
/tmp/dbg_fw.bin -o /tmp/dbg_token.xml generate_token
```

To create a CRDT token with nested Retimer:

```
# mlxtokengenerator -d mt54004_pciconf0_cable_1 -k CRDT -t LinkX
-debug_fw /tmp/dbg_fw.bin --nested_debug_fw
/tmp/retimer_dbg_fw.bin --nested_token -o /tmp/dbg_token.xml
generate_token
```

To create challenge based tokens with blob option

```
# mlxtokengenerator -b <bllob_file> -k CRCS -t HCA -o <out.xml>
generate_token
```

mlxdpa – DPA Applications Sign Tool

The mlxdpa tool allows the user to sign DPA applications, which are given to the tool as part of a Host ELF file.

It also supports creation, signing, and removal of single applications.

In addition, mlxdpa allows the user to add or remove certificates from the DPA device — this is done by creating certificate containers and signing them.

The tool generates the signatures using a provided private key PEM file.

Tool Requirements:

- Supported operating systems: Linux
- Supported platforms: x86-64, arm64

mlxdpa Synopsis

Sign Host ELF using PEM file

```
# mlxdpa --host_elf <ELF file> --cert_chain <certificate chain> -
-private_key <key .pem file> --output_file <output file path>
sign_dpa_apps
```

Create upload container for single app

```
mlxdpa -s /tmp/singleApp.elf --life_cycle_priority OEM  -m
/tmp/appmetadata.yaml --manifest /tmp/manifest.bin -o
```

```
/tmp/single_app.bin create_single_dpa_app
```

Sign upload container for single app using PEM file

```
mlxdpa -s /tmp/single_app.bin -c /tmp/chain.cert -p
/tmp/p_key.pem [--cert_chain_count 5] --life_cycle_priority OEM -
o /tmp/signed_single_app.bin sign_single_dpa_app
```

Query manifest from single elf

```
mlxdpa -s /tmp/singleApp.elf -o /tmp/manifest.bin query_manifest
```

Create Dpa app removal container

```
mlxdpa --dpa_app_uuid 7c0ab0fc-082e-11ee-bd9d-e43d1a1f06ae -o
/tmp/dpa_app_removal_container.bin --life_cycle_priority OEM
create_dpa_app_removal
```

Sign Dpa app removal container

```
mlxdpa --dpa_app_removal_container
/tmp/dpa_app_removal_container.bin --keypair_uuid 3c8f46b2-159f-
11ee-9ac4-e43d1a1f06ae -p /tmp/p_key.pem-o
/tmp/signed_dpa_app_removal_container.bin --life_cycle_priority
OEM sign_dpa_app_removal
```

Where:

-e|--host_elf	Path to the Host ELF file containing DPA applications
-c|--cert_chain	Path to a certificate chain file to embed in the crypto data
-p|--private_key	Path to a private key PEM file for signature generation
-o|--output_file	Path to output signed Host ELF
-h|--help	Show help message
-v|--version	Show tool version
--cert_chain_count <Hex number>	Number of certificates in the provided certificate chain
--dpa_app_removal_container <Path>	Path to a dpa app removal container to sign
--manifest <Manifest>	Path to the manifest file
-m|--app_metadata <App Metadata>	Path to the app metadata yaml file
-s|--single_app <Single App>	Path to the single app file

Creating a Certificate Container

Container for adding a certificate:

```
mlxdpa --cert_container_type add -c <.DER formatted certificate>
-o <output path> --life_cycle_priority <Nvidia,OEM,User>
create_cert_container
```

Container for removing a certificate:

```
mlxdpa --cert_container_type remove [--cert_uuid <uuid of the
certificate for removal>] [--remove_all_certs] -o <output path> --
life_cycle_priority <Nvidia,OEM,User> create_cert_container
```

Create a certificate upload container with the keep_sig flag

```
mlxdpa --cert_container_type add -c /tmp/cert.der -o
/tmp/cert_container.bin --life_cycle_priority OEM --keep_sig
create_cert_container
```

Create certificate upload container with nvidia_signed_oem flag

```
mlxdpa --cert_container_type add -c /tmp/cert.der -o
/tmp/cert_container.bin --nvidia_signed_oem create_cert_container
```

Signing a Certificate Container

Container for adding a certificate:

```
mlxdpa --cert_container <container> -p <private key pem file> --
keypair_uuid <uuid> --cert_uuid <uuid> --life_cycle_priority
<Nvidia,OEM,User> -o <output path> sign_cert_container
```

Container for removing a certificate:

```
mlxdpa --cert_container <container> -p <private key pem file> --
keypair_uuid <uuid> --life_cycle_priority <Nvidia,OEM,User> -o
<output path> sign_cert_container
```

Where:

--cert_container	Path to a certificate container to sign
--cert_container_type <Add/Remove>	Type of a certificate container to create

-c|--certificate	Path to a .DER formatted certificate
--keypair_uuid	Key-pair UUID of the private key used for signing
--cert_uuid	Time base UUID generated right before signing
--remove_all_certs	Remove all CA Certificates, provide with the sign_cert_remove command
--life_cycle_priority <Nvidia, OEM, User>	Life-cycle priority of a requested certificate container
-o|--output_file	Path to an output file
-p|--private_key	Path to a private key PEM file for signature generation
--nvidia_signed_oem	NVIDIA signed OEM certificate
-k|--keep_sig	The whole certificate container will be kept

mlxcableimgen - Cable Firmware Image Wrapper Generation Tool

The specifications of a compatible cable hardware must be passed to the tool to create the image wrapper.

mlxcableimgen Synopsis

```
mlxcableimgen -d <OUI> -p <vendor pn> -r <vendor rev> -w <vendor
hw major> -m <vendor fw major> -n <vendor fw minor> -b <vendor fw
build> -i <fw image> [-j <vendor data>] -o <output path>
generate_image
```

Where:

-d|--vendor_ieee_id	Vendor IEEE ID in Hex (6 characters)
-p|--vendor_pn	Part number provided by vendor in ASCII (16 characters)
-r|--vendor_rev	Revision level for part number provided by vendor in ASCII (2 characters)

-w|--vendor_hw_major	Module hardware major revision in Hex (2 characters)
-m|--vendor_fw_major	Module firmware major revision in Hex (2 characters)
-n|--vendor_fw_minor	Module firmware minor revision in Hex (2 characters)
-b|--vendor_fw_build	Module firmware build revision in Hex (4 characters)
-i|--image_path	Path to FW image file
-j|--vendor_data_path	Path to vendor data file (optional)
-o|--output_path	Path to output image file

nvredfish

The nvredfish tool enables the user to query the BMC firmware and update the BMC firmware.

Users can perform firmware update to BMC (for the update to take effect, a full AC cycle of the board is needed).

Note

For the update to take effect, a full AC cycle of the board is needed.

Note

Initial FW burn is outside the scope of this feature.

Note

For the nvredfish tool to be installed as part of the MFT package, the user must provide `--with-nvredfish` argument to the install script.

User Flow

1. Generate token (not MFT related)
2. Generate mst device representing the BMC:

   ```
   mst redfish add <bmc ip> <token>
   ```

3. Query:

   ```
   nvredfish -d <> query
   ```

4. Burn the image:

   ```
   nvredfish -d <> -p <.fwpkg> burn
   ```

After the burn command, for the firmware update to take effect, an AC cycle (physically disconnecting and then reconnecting the system’s main power) of the board is needed.

Debug Utilities

This section contains:

- fwtrace Utility
- mlxtrace Utility
- mlxptrace Utility
- mstdump Utility
- mlxi2c Utility
- i2c Utility

- mget_temp Utility
- mlxdump Utility
- mlxmcg Utility
- pckt_drop Utility
- mlxuptime Utility
- wqdump Utility
- mlxmdio Utility
- mlxreg Utility
- mlxlink Utility
- mlxfwstress Utility
- resourcedump Utility
- resourceparse Utility
- stedump Utility

fwtrace Utility

The fwtrace utility extracts and prints trace messages generated by the firmware running on 5th generation (Group II) devices iRISCs.

These trace messages inform developers of software drivers about internal status, events, critical errors, etc. Trace messages generated by iRISCs are stored in the trace buffer. The trace buffer is located in host memory. The tool also supports mem free mode where it uses a device internal small buffer.

By default, the firmware does not print trace messages. Please contact your FAE for more details on how to enable firmware tracing.

Note

When using secure firmware, the user needs to validate that the value "1" is set to /sys/kernel/debug/tracing/events/mlx5/fw_tracer/enable.

As of MFT v4.30.0, fwtrace can extract events from secure devices (for supported Firmware versions) using mlxptrace utility and without the need of mlx5 driver. fwtrace will automatically use the new capabilities if the device supports it.

Note

Memory mode on 5th generation (Group II) devices is supported only by PCI mst devices.

Note

For the tool to properly work with Inband devices, both the MFT and the Firmware must be updated to the latest (MFT version 4.18.0 and above, and Firmware version XX.32.1xxx and above).

fwtrace Usage

1. Start the mst driver (mst start or mst restart)
2. Enter the following command:

```
# fwtrace [options...]
```

where

-h|--help	Print this help message and exit

-d|--device	mst device name
-f|--fw_strings	Fw strings db file containing the FW strings
--tracer_mode	Tracer mode [FIFO | MEM]
--real_ts	Print real timestamps in [hh:mm:ss:nsec] format
-i|--irisc	iRISC name (See below for full list of irisc names)
-t|--tile	Allow the user to specify which tile events he would like to see. # "1.all" will enable risc1 in all tiles. # "1" will be treated like "1.all" # "all" will be treated like all.all # "all.1" will enable all iriscs in tile 1 # .1 is invalid input.
-a|--apu	If enabled, tool will show APU events.
--include_phy_uc	If enabled, tool will also show phy_uc events related to the enabled riscs (main / tiles)
--keep_cfg	If enabled, the FWtrace will not delete the temporary cfg file (if one was created during the current run)
-s|--stream	Run in streaming mode
-c|--cfg	Hardware tracer events cfg file
-n|--snapshot	Take events snapshot - this assumes previous FW configurations
-S|--buf_size	Hardware tracer MEM buffer size in [MB]
-p|--dump	Dump file name
--raw_dump	Used with --dump flag to inform the tool the dump input type is .dump, which is coming from reasource_dump, and not .trc type that come from tracer tool
-m|--mask	Trace class mask, use "+" to enable multiple classes or use integer format, e.g: -m class1+class2+... or 0xff00ff00
-l|--level	Trace level
--log_delay	Firmware tracer log delay in uSec
v|--version	Print tool's version and exit
--gvmi	Global virtual machine interface
--ignore_old_events	Ignore collecting old events
--mem_access	Memory access method: OB_GW, VMEM, UDRIVER

--keep_running	Keep the hardware tracer unit running after exit
--config_only	Configure tracer and exit
--fw_cfg_only	Skip HW config and only configure FW events (default=off)

Device Specific Info:

To view device specific information, run `"fwtrace -d <mst_dev> -h".`

Example:

```
fwtrace -d /dev/mst/mt4123_pciconf0 -i all -s -l 0 -m 0xFFFFFFFF
-I- Found FW string db cache file, going to use it
Got ownership successfully!
mlxtrace -d /dev/mst/mt4123_pciconf0 -m MEM --skip_ownership -c
/tmp/itrace_2946580.cfg -S
-I-     Tracer Configuration:
-I-     =====================
-I-     Mode                                        : Collector
-I-     Activation Mode                             : Memory Mode
-I-     Memory Access Method                        : NA
-I-     Configuration File Path                     :
/tmp/itrace_2946580.cfg
-I-     Output file (Trace File) Path               : mlxtrace.trc
-I-     User Buffer Size                            : NAMBytes
-I-     Use Stream Mode                             : YES
-I-     Configure Only                              : NO
-I-     Only Snapshot (Skip Configuration Stage)    : NO
-I-     Ignore old events                           : NO
-I-     Continuous fill                             : NO
-I-     Print timestamp in hh:mm:ss:nsec format : NO
-I-     Output file for streaming                  : STDOUT
-I-     Delay between samples                       : 0usec
-I-     Stop tracer after exit                      : YES
-I-     ===============================================
-I-     Tracer ownership is already taken or skipped
```

```
-I-     Device is: ConnectX6
-I-     Configuring Tracer...
-I-     Invalidating kernel buffer... (Press ^C to skip)
-I-     Done
-I-     Tracer was configured successfully
Device frequency: 430MHz
-I-     Starting event streaming...
#time stamp, unreliable, lost, missing TS, event name
Reading new events...
512458354925     I3              in pll_management_handler. freq_in
= 3.
512458355201     I3              in pll_management_handler. set pll
0 in center pll 0 to state 0.
512458355749     I3              in pll_management_handler. freq_in
= 3.
512458356030     I3              in pll_management_handler. set pll
1 in center pll 0 to state 0.
512458356583     I3              in pll_management_handler. freq_in
= 3.
512458356865     I3              in pll_management_handler. set pll
2 in center pll 0 to state 3. .
.
.
```

mlxtrace Utility

The mlxtrace utility is used to configure and extract HW events generated by different units in NVIDIA devices. The utility generates a dump ".trc" file which contains HW events that assist us with debug, troubleshooting and performance analysis. Events can be stored in host memory if driver is up or in a small on-chip buffer (always available) depending on the utility running mode. In order to run the utility it's required to have a configuration file first, this file should be provided by the NVIDIA representative.

A dump file "mlxtrace.trc" will be generated by end of run (file name can be controlled by "-o" flag), this file should be sent to the NVIDIA representative for further diagnostics/troubleshooting.

> **Note**
>
> Memory mode on 5th generation (Group II) devices is supported only by PCI mst devices. Memory mode is supported in Windows, as well as in Linux.

> **Note**
>
> For the tool to properly work with Inband devices, both the MFT and the Firmware must be updated to the latest (MFT version 4.18.0 and above, and Firmware version XX.32.1xxx and above).

mlxtrace Usage

1. The mst driver must be started prior to running the mlxtrace tool.
2. For MEM buffer mode driver must be "loaded" also.
3. Enter the following command:

```
mlxtrace [options]
```

Options

-h, --help	Print help and exit
-v, --version	Print version (default=off)
-p, --parse	Move to parser mode (default=off)

Mode: CollectMode

-d, --device=MstDev	Mst device

-m, --mode=Mode	Activation mode: FIFO - HW BUFFER , MEM - KERNEL BUFFER (possible values="FIFO", "MEM")
-a, --mem_access=MemMethod	Memory access method: OB_GW, VMEM, UDRIVER, PERFMON (possible values="OB_GW"/ "VMEM"/ "UDRIVER" "PERFMON")
--gvmi	Global virtual machine interface
-c, --cfg=CfgFile	Mlxtrace configuration file
-o, --trc_file=TrcPath	Output TRC file path (default=`mlxtrace.trc')
-C, --config_only	Configure tracer and exit (default=off)
-b, --backpressure	Enable "Allow_backpressure" for all supported hardware events (default=off)
-n, --snapshot	Take events snapshot - This assumes previous run with --config_only (default=off)
-s, --buf_size=BufSize	User buffer size [MB] (default=` 1')
-S, --stream	Don't save events to file parse it immediately (default=off)
--ignore_old_events	Ignore collecting old events in MEM mode (default=off)
-g, --continuous_fill	Do not stop recording (stopping only with ^C), keep filling user's buffer cyclically (default=off)
--sample_delay=Delay	Delay between samples when polling new events in [usec] (default=`0')
--keep_running	Keep the HW tracer unit running after exit (default=off)
--enable_limiting_every_chunk	Limit the HW tracer after reading every chunk (default=off)
--fw_cfg_only	Skip HW config and only configure FW events (default=off)
--skip_ownership	Skip taking ownership (default=off)

Mode: ParseMode

-i, --input=TrcFile	Input file (default=`mlxtrace.trc')
--csv_mode	Enable csv output format (default=off)
--print_ts	Print timestamp events (default=off)

-r, --real_ts	Print real timestamps in [hh:mm:ss.nsec] format (default=off)
--print_raw	Print event bytes in each line header (default=off)
--ts_format=format	Choose printed TS format hex/dec (possible values="hex", "dec" default=`dec')
--print_delta	Enable printing delta between events (in cycles) (default=off)
-f, --print_file=FilePath	Print parsed event to the given file and not to stdout
--enable_db_check	Enable events DB checks (default=off)

Examples:

Choose the suitable .cfg file depending on the device you are using, and run the following command to generate a .trc file:

```
# mlxtrace -d /dev/mst/mt4123_pciconf0 -c ConnectX6.cfg -m MEM -o
ConnectX6.trc
```

To generate a .trc file with a maximal size of 100 MB, run the following command:

```
# mlxtrace -d /dev/mst/mt4123_pciconf0 -c ConnectX6.cfg -m MEM -s
100 -o ConnectX6.trc
```

mlxptrace Utility

The mlxptrace utility is used to configure and extract FW events generated by different units in NVIDIA devices.

This utility was added in MFT v4.30 to provide solutions for supported secured devices with the need of debug token.

- The utility is only supported in Linux environment
- It is recommended to run the tool via fwtrace and not directly from command line

- the tool has similar interface to the mlxtrace and same .cfg files can be used for both tools. (for more information please review the mlxtrace utility)

mlxptrace Usage

1. The mst driver must be started prior to running the mlxptrace tool.
2. Enter the following command:

   ```
   mlxptrace [options]
   ```

Options:

-h, --help	Print help and exit
-p, --parse	Move to parser mode (default=off)
-d, --device=MstDev	Mst device
-m, --mode=Mode	Currently only FIFO mode is supported (possible values are "FIFO", "MEM" for compatibility with mlxtrace).
--gvmi	Global virtual machine interface
-c, --cfg=CfgFile	mlxtrace configuration file
-o, --trc_file=TrcPath	Output TRC file path (default=`mlxtrace.trc')
-C, --config_only	Configure tracer and exit (default=off)
-n, --snapshot	Take events snapshot - This assumes previous run with --config_only (default=off)
-s, --buf_size=BufSize	User buffer size [MB] (default=` 1')
-S, --stream	Don't save events to file parse it immediately (default=off)
--ignore_old_events	Ignore collecting old events in MEM mode (default=off)
-g, --continuous_fill	Do not stop recording (stopping only with ^C), keep filling user's buffer cyclically (default=off)
--keep_running	Keep the HW tracer unit running after exit (default=off)
--fw_cfg_only	Skip HW config and only configure FW events (default=off)
--skip_ownership	Skip taking ownership (default=off)

-i, --input=TrcFile	Input file (default=`mlxtrace.trc')
--raw_dump	When set, the parse input file will be considered as raw dump taken from resource dump instead of .trc file. Must be used with --cfg and --parse flags.
--csv_mode	Enable csv output format (default=off)
--print_ts	Print timestamp events (default=off)
--print_raw	Print event bytes in each line header (default=off)
--ts_format	Switch printed TS format to hex, default=dec
--print_delta	Enable printing delta between events (in cycles) (default=off)

mstdump Utility

The mstdump utility dumps device internal configuration registers. The dump file is used by the Support team for hardware troubleshooting purposes. It can be applied on all NVIDIA devices.

Note

For the tool to properly work with Inband devices, both the MFT and the Firmware must be updated to the latest (MFT v4.18.0 & firmware vXX.32.1014).

Note

If ConnectX-4 adapter card is used as an Inband device, for the tool to work properly, you need to use MFT 4.17.0.

mstdump Usage

To run mstdump:

```
# mstdump [-full] <mst device> > <dump file>_FW_VERSION
```

where the `-full` flag dumps all internal registers

positional arguments:

device	The device name

optional arguments:

-h, --help	show help message and exit
-v, -version, --version	show program's version number and exit
-full, --full	Dump more expanded list of addresses
-ignore_fail, --ignore_fail	Continue dumipng, even if some addresses fails
-c CSV, -csv CSV, --csv CSV	Database path
--cause address.offset	Specify address and offset
--i2c_secondary I2C_SECONDARY	I2C secondary [0-127]

Example:

```
[root@mymach]# mstdump /dev/mst/mt4099_pci_cr0 >
mt4099_12_16_2600.dmp
```

This dumps the internal configuration data of the device into the mt4099.dmp file.

mlxi2c Utility

The mlxi2c utility provides a way to route the I2C bus to 4th generation (Group I) switches.

mlxi2c Usage

The mst driver must be started prior to running mlxi2c.

To start mlxi2c:

1. Start the mst driver (mst start or mst restart). Note: This step in not required in Windows.

2. Run mlxi2c with the following command line syntax:

```
# mlxi2c [switches...] <command> [parameters...]
```

Switches Options

-d <device>	mst i2c device name default: "/dev/mst/mtusb-1" Affected commands: all
-h	Print this help information
-s, --i2c-secondary <address>	Change the I2C secondary address.
-v	Print version and exit

Commands

p <i2c_component>	Route the i2c path to the indicated i2c component
scan	Scan the i2c slave addresses

Example:

Display the addresses of all I2C-accessible devices:

```
# mlxi2c -d /dev/mst/mtusb-1 scan
```

i2c Utility

The i2c utility provides low level access to the I2C bus on any NVIDIA switch platform, enabling the user to read or write data.

i2c Usage (Advanced Users)

The mst driver must be started prior to running i2c tool.

To start i2c:

1. Start the mst driver (mst start or mst restart). Note: This step in not required in Windows.

2. Run i2c with the following command line syntax:

```
# i2c [OPTIONS] <device> <cmd> <i2c_addr> <addr> [<data>]
```

where:

-h	Prints this message.
-a <addr_width>	Sets address width (in bytes) to the specified value. May be 0, 1, 2 or 4. Default: 1.
-d <data_width>	Sets data width (in bytes) to the specified value. May be 1, 2 or 4s. Default is 1.
-x <data_len>	Presents each byte of data as two hexadecimal digits (such as 013C20343B). Note that this option is mutually exclusive with the "-d" option.

The remaining parameters are:

<device>	Valid mst device.
<cmd>	Command. May be "r[ead]" or "w[rite]".
<i2c_addr>	I2C slave address.
<addr>	Address (of length addr_width) inside I2C target device to read/write operation. Note that the <addr> value is ignored if <addr_witdh> = 0.
<data>	Data (bytes of length data_width) to write to target device.

ⓘ **Note**

All parameters are interpreted as hexadecimal values.

Examples:

Read two bytes from address 0 of target I2C slave address 0x56:

```
# i2c -a 2 -d 2 /dev/mst/mtusb-1 r 0x56 0x00
0000
```

Write two bytes to the address above then read them:

```
# i2c -a 2 -d 2 /dev/mst/mtusb-1 w 0x56 0x00 0x1234
# i2c -a 2 -d 2 /dev/mst/mtusb-1 r 0x56 0x00
3412
```

Read (as separate) 16 bytes in hexadecimal format starting from address 0 of the target device above:

```
# i2c -a 2 -x 16 /dev/mst/mtusb-1 r 0x56 0x00
12340000000000000000000000000000
```

Exit Return Values

The following exit values are returned:

- 0 - successful completion
- >0 - an error occurred

mget_temp Utility

The mget_temp utility reads the hardware temperature from NVIDIA devices with temperature sensors (all NVIDIA devices) and prints the result in Celsius degrees.

mget_temp Usage

To run mget_temp:

```
# mget_temp [OPTIONS]
```

where:

-h	Print the help message.
-d <dev>	mst device name.
--version	Display version info.
-s <i2c-secondary>	I^2C secondary address
-v	print a table of all thermal diode data

Example on how to read a device temperature:

```
# mget_temp -d /dev/mst/SW_MT51000_0002c903007e76a0_lid-0x0002
```

> **Note**
>
> mget_temp utility reads the IC temperature, it does not support reading temperature from peripheral sensors on the board.

mlxdump Utility

The mlxdump utility dumps device internal configuration data and other internal data (such as counters, state machines).

The data can be used for hardware troubleshooting. It can be applied to all NVIDIA devices.

The tool has 3 run modes: [fast | normal | full] while the default is "fast", the "full" mode dumps all available data but might run slower than normal and fast modes.

The tool also can dump only flow steering information using the fsdump sub-command (See example below). The fsdump sub-command has the flag --type to specify the type of the flow steering: STE or FT or All.

The tool can dump only mstdump information using the mstdump sub-command (see example below). The mstdump sub-command has multiple flags: --full, i2c_slave, cause_addr and cause_offset which enable the user to run with the needed parameters.

mlxdump Usage

The mst driver must be started prior to running mlxdump tool.

```
mlxdump OPTION <command> [COMMAND OPTIONS] [-d|--device
MstDevice] [-h|--help] [-v|--version]
```

where:

-d|--device MstDevice	mst device name
-h|--help	Show help message and exit
-v|--version	Show version and exit
mstdump	Read mstdump information
fsdump	Read Flow Steering information. **Note:** Reading flow steering information is supported in ConnectX-4 and above adapter cards.
snapshot	Dump everything

> **Note**
>
> To view <command> related options please run: "mlxdump OPTION <command> -h"

Examples:

To generate "mlxdump.udmp" while running in fast mode:

```
# mlxdump -d /dev/mst/mt4117_pciconf0 snapshot
```

To generate "mlxdump.udmp" while running in full mode:

```
# mlxdump -d /dev/mst/mt4117_pciconf0 snapshot -m full
```

To generate "mlxdump_13_1_2013.udmp" while running in normal mode:

```
# mlxdump -d /dev/mst/mt4117_pciconf0 snapshot -m normal -o
mlxdump_13_1_2013.udmp
```

To generate flow steering information:

```
# mlxdump -d /dev/mst/mt4117_pciconf0 fsdump --type=All --gvmi=0
```

To generate mstdump information::

```
# mlxdump -d /dev/mst/mt4119_pciconf0 mstdump
```

mlxmcg Utility

The mlxmcg tool displays the current multicast groups and flow steering rules configured in the device. Target users: Developers of Flow Steering aware applications.

This tool dumps the internal steering table which is used by the device to steer Ethernet packets and Multicast IB packets to the correct destination QPs.

Each line in the table shows a single filter and a list of destination QPs. Packets that match the filter are steered to the list of destination QPs.

Note

- mlxmcg is not supported against In-band device.
- mlxmcg is supported in ConnectX-3/ConnectX-3 Pro only.

mlxmcg Usage

The mst driver must be started prior to running mlxmcg tool. To start mlxmcg:

1. [Optional for Windows OSs] Start the mst driver (mst start or mst restart).
2. Enter an mlxmcg command that complies with the following command syntax:

```
# mlxmcg [OPTIONS]
```

where:

-h, --help	Show this help message and exit

-d DEV, --dev=DEV	mst device to use, required
-f FILE, --file=FILE	MCG dump file to use (for debug). Used as input - no need for a device.
-p PARAMS, --params=PARAMS	Mcg params, (MCG_ENTRY_SIZE, HASH_TABLE_SIZE, MCG_TABLE_SIZE), default is (64, 32768, 65536)
-q, --quiet	Do not print progress messages to stderr
-v, --version	Print tool version
-c, --hopcount	Add hopCount column
-a, --advanced	Show all rules

This will display all the current multicast groups and flow steering rules configured in the device.

Example:

```
Command : mlxmcg -d /dev/mst/mt4099_pci_cr0
MCG table size: 64 K entries, Hash size: 32 K entries, Entry size:
64 B
Progress: HHHHHLLLL
Bucket Index ID Prio Proto DQP Port VLAN MAC SIP DIP I-MAC I-VLAN
VNI L4 SPort DPort Next QPs
0 0 0 0 all -- 2 -- -- -- -- -- -- -- -- -- -- 1009c 11
af9 fee0 0 5000 IPv6 -- ff0e:0000:0000:0000:0000:0000:e000:0001 - -- --
-- -- -- 8012 SB
e3f e3f 0 5000 L2 -- 2 -- 01:80:c2:00:00:0e -- -- - -- -- -- -- --
8012 40048
1139 1139 0 5000 L2 -- 2 -- 01:00:5e:00:00:01 -- -- - -- -- -- -- --
8014 40048
26fc 26fc 0 5000 L2 -- 2 -- ff:ff:ff:ff:ff:ff -- -- - -- -- -- --
-- 8018 40048
2a3e 2a3e 0 5000 L2 -- 2 -- 33:33:00:00:00:01 -- -- -- -- -- -- -- --
801a 40048
4000 4000 0 0 all -- 2 -- -- -- -- -- -- -- -- -- -- 1009c 10
45d7 45d7 0 5000 L2 -- 1 -- 01:00:5e:00:00:fb -- -- -- -- -- -- -- -
- 8000 4004a
```

```
4af9 fef8 0 5000 IPv6 -- ff0e:0000:0000:0000:0000:0000:e000:0001 -- --
-- -- -- 8002 SB
4e3f 4e3f 0 5000 L2 -- 1 -- 01:80:c2:00:00:0e -- -- -- -- -- -- -- -
- 8002 4004a
5139 5139 0 5000 L2 -- 1 -- 01:00:5e:00:00:01 -- -- -- -- -- -- -- --
8004 4004a
66fc 66fc 0 5000 L2 -- 1 -- ff:ff:ff:ff:ff:ff -- -- -- -- -- -- --
-- 8008 4004a
6a3e 6a3e 0 5000 L2 -- 1 -- 33:33:00:00:00:01 -- -- -- -- -- -- -- --
800a 4004a
734b 734b 0 5000 L2 -- 1 -- 33:33:e0:00:00:01 -- -- -- -- -- -- -- -
- 800c 4004a
Duplicated MCGS: Count
1000 1000 0 5000 L2 -- 2 -- 00:02:c9:00:00:02 -- -- -- -- -- -- -- --
8015 40048 2048
4002 4002 0 5000 L2 -- 1 -- 00:02:c9:00:00:01 -- -- -- -- -- -- -- --
8001 4004a 2046
16 Unique rules, 4108 Total
Index QPs
======================================================================
fee0 40054 40055 40056 40057 40058 40059 4005a 4005b 4005c 4005d 4005e
4005f 40060
40061 40062 40063 40064 40065 40066 40067 40068 40069 4006a 4006b 4006c
4006d
4006e 4006f 40070 40071 40072 40073 40074 40075 40076 40077 40078 40079
4007a
4007b
======================================================================
fef8 40054 40055 40056 40057 40058 40059 4005a 4005b 4005c 4005d 4005e
4005f 40060
40061 40062 40063 40064 40065 40066 40067 40068 40069 4006a 4006b 4006c
4006d
4006e 4006f 40070 40071 40072 40073 40074 40075 40076 40077 40078 40079
4007a
4007b
======================================================================
```

pckt_drop Utility

The pckt_drop utility corrupts the next transmitted packet from a ConnectX family adapter port.

pckt_drop Usage

Run the pckt_drop with the following command line syntax:

-d, --device	Specify the mst device to configure. (**Required**.)
-h, --help	Print this help screen and exit.
-m, --mode	Specify operating mode. Supported modes are: **EDP**: Inserts error on next transmitted data packet. (Default: EDP)
-p, --port	Select which port to configure. Use `1'/`2' for port1/port2, respectively, or `b' for both. (Default: b)
-v, --version	Print the application version and exit.

Example:

```
# pckt_drop -d /dev/mst/mt4117_pciconf0 -p 1
```

The example above shows how to use the pckt_drop to corrupt a packet from port 1.

mlxuptime Utility

The mlxuptime is a firmware which prints NVIDIA devices' up time and measured/configured frequency.

mlxuptime Usage

```
mlxuptime [options]
```

where:

-d <dev> --device	Mst device name
-s <time> --sample	Sampling interval for measuring frequency (default: 1 [sec])
-h --help	Print help and exit
-v --version	Print tool version and exit
-f, --force_sample	Force sampling interval for measuring frequency. Default: Reading up time from device.

Examples:

Print all info:

```
# mlxuptime -d /dev/mst/mt4117_pciconf0
Measured core frequency       : 427.099818 MHz
Device up time                : 10:01:20.456344 [h:m:s.usec]
```

wqdump Utility

The wqdump utility dumps device internal work queues. A work queue is an object containing a Queue Pair Context (QPC) which contains control information required by the device to execute I/O operations on that QP, and a work queue buffer which is a virtually-contiguous memory buffer allocated when creating the QP.

The dumped data can be used for hardware troubleshooting. It can be applied on ConnectX adapter cards family and Connect-IB adapter devices.

Note

wqdump on ConnectX-3 and ConnectX-3 Pro is not supported against in-band devices.

wqdump Usage

The mst driver must be started prior to running the wqdump utility. To start the wqdump utility:

1. Start the mst driver (mst start or mst restart).
2. Run wqdump:

```
# WQDump <-d|--device DeviceName> <--source ContextType> [--gvmi Gvmi] [--qp ContextNumber]
<--dump DumpType> [--fi StartIndex] [--num NumberOfItems] [--format Format]
[--address Address] [--size Size] [-v|--version] [-h|--help]
[--clear_semaphore] [--gw_access]
```

where:

--d|--device DeviceName	Device name
--source ContextType	Type of context to dump. Options are: Snd, Rcv, Cmp, Srq, Eqe, Connect-X3/Pro: MCG,5th generation devices: MKC, SXDC, FullQpConnectX-5: CMAS_QP_WQE, CMAS_QP_SWQ, CMAS_SRQ_WQE, CMAS_CQ_BUFF, CMAS_QP_DBR, CMAS_QP_SDB, CMAS_SRQ_DB, CMAS_CQ_DBR, CMAS_CQ_ARM, CMAS_EQ_BUFF, CMAS_TAG_MATCH.CMAS_INLINE
--gvmi Gvmi	Guest VM ID (5th generation devices)
--qp ContextNumber	Context number to dump
--dump DumpType	Dump Type. Options are: WQ, QP, WQ_QP, ALL_QPC, ALL_VALID_QPNS, ICM

--fi StartIndex	Index of first element to dump, (Default:0)
--num NumberOfItems	Number of elements to dump from buffer, (Default: keep reading)
--format Format	Output format: options are : text, raw, dw, (Default: text)
--address Address	Memory Address
--size Size	Memory size in bytes
-v|--version	Show tool version and exit
-h|--help	Show usage
--clear_semaphore	Force clear semaphore
--gw_access	Force get QPC by GW access (Connect-X 3/Pro)

4th Generation Device Examples:

Print all valid qpns

The example below will dump all valid qpns of type mcg context.

```
# wqdump -d /dev/mst/mt4099_pci_cr0 --source mcg --dump
ALL_VALID_QPNS
```

Dump mcg qp

The example below will dump mcg context number 0x10.

```
# wqdump -d /dev/mst/mt4099_pci_cr0 --source mcg --dump QP -qp
0x10
```

Dump other qpns

The example below will dump snd context number 0x10 in a raw format.

```
# wqdump -d /dev/mst/mt4099_pci_cr0 --source snd --dump QP -qp 0x10 --format raw
```

Dump wq

The example below will dump send work queue buffer number 0x42.

```
# wqdump -d /dev/mst/mt4099_pci_cr0 --source snd --dump wq -qp 0x42
```

Dump mcg qp by GW access

The example below will dump mcg context number 0x10 by GW access.

```
# wqdump -d /dev/mst/mt4103_pci_cr0 --source mcg --dump QP -qp 0x10 --gw_access
```

5th Generation Device Examples:

FullQp: The QP context of the Rcv and Snd with the common part. Note, FullQp does not have dump as WQ.

- Get opened contexts from the first 20 indexes:

```
# wqdump -d /dev/mst/mt4117_pciconf0 --source FullQp --dump ALL_VALID_QPNS --num 20
Numbers of valid contexts (in the range 0x0 - 0x13):
index 0x00000003
index 0x00000006
index 0x00000007
index 0x0000000b
index 0x0000000c
```

```
index 0x00000013
-----------------------------------------
Number of valid contexts: 6
-----------------------------------------
```

• Show the QP Context (RAW):

```
# wqdump -d /dev/mst/mt4117_pciconf0 --source FullQp --dump QP --
qp 0x0000000b --format raw
== Common Part (Not Connected) ==
0. 80000000 0000001e 05000000 00000000
1. 70000000 00000000 00000000 00000000
2. 0000000b 00ffffff 00000000 00000000
3. 00000000 00000000 00000000 80010000
--------------------------
Send Qpc gvmi 0000 QP Index 0000000b
0. 80000000 0000001e 05000000 00000000
1. 70000000 00000000 00000000 00000000
2. 0000000b 00ffffff 00000000 00000000
3. 00000000 00000000 00000000 80010000
--------------------------
== Responder Part (Not Connected (mac)) ==
Recv Qpc gvmi 0000 QP Index 0000000b
0. b8eccd02 00000000 d8f5cd02 00000000
1. d0010000 00000000 40000000 00000000
2. e0e03903 00000000 f0e03903 00000000
3. 00000000 00000000 00000000 00000000
--------------------------
```

SRQ

• Opened QPs:

```
# wqdump -d /dev/mst/mt4115_pciconf0 --source Srq --dump
all_valid_qpns
Numbers of valid contexts (in the range 0x0 - 0xffffff):
gvmi 0x0000 index 0x00000060
gvmi 0x0000 index 0x00000061
gvmi 0x0000 index 0x00000066
```

- Dump WQs:

```
0x00000066# wqdump -d /dev/mst/mt4117_pciconf0 --source Srq --dump
wq --qp 0x60
[Element Index 0]
-------------- SRQ Next ------------------
next_wqe_index : 0x1
signature : 0x0
-------------- scatter entry (0) ---------------
byte_count : 0x0
wqe_inline : 0x0
local_key : 0x0
local_address_63_32 : 0x0
local_address_31_0 : 0x0
[Element Index 0x1]
-------------- SRQ Next ------------------
next_wqe_index : 0x0
signature : 0x0
-------------- scatter entry (0) ---------------
byte_count : 0x0
wqe_inline : 0x0
local_key : 0x0
local_address_63_32 : 0x0
local_address_31_0 : 0x0
```

CMAS

• Opened contexts from some CMAS type (CMAS_EQ_BUFF for ex):

```
#wqdump -d /dev/mst/mt4119_pciconf0 --source CMAS_EQ_BUFF --dump
ALL_VALID_QPNS
Numbers of valid contexts (in the range 0x0 - 0xffffff):
gvmi 0x0000 index 0x00000002
gvmi 0x0000 index 0x00000010
gvmi 0x0000 index 0x00000011
gvmi 0x0000 index 0x00000012
gvmi 0x0000 index 0x00000013
gvmi 0x0000 index 0x00000014
gvmi 0x0000 index 0x00000015
```

• Dump raw data:

```
# wqdump -d /dev/mst/mt4119_pciconf0 --source CMAS_EQ_BUFF --dump
QP --qp 0x15 --format raw
CMAS gvmi 0000 CMAS Index 00000015
0. 80000002 00000000 00000000 e4f80000
1. 00000000 00000000 00000000 00000000
2. 00000000 00000000 00000000 00000000
3. 00000000 00000000 00000000 00000000
--------------------------
```

mlxmdio Utility

The mlxmdio tool is used to read/write MDIO registers (Clause 45) on Boards with externally managed PHY.

mlxmdio Usage

To run mlxmdio, use the following line:

```
# mlxmdio -d mst_dev -m phy_addr:dev_addr -g mdio_gw -a
addr[:data] [-r size] [-w input_file]
```

where:

-d <device>	mst device
-m <mdio_id>	The mdio id of the target device in phy_addr:dev_addr format.
-a <addr[:data]>	Access a single MDIO reg. If data is specified, the reg is written, Otherwise, it is read. Addr and data should be in hex format.
-g <mdio_gw>	Select which mdio gateway <0..10> to use.
-c <clause>	Select which clause to use: • 22: clause 22. • 45: clause 45 (Default).
-r <size>	Read a block of <size> 16-bit words (max size 64)
-w <input_file>	Write a block from <input_file>. Every line of input file should be addr:data in hex.
-h	Show usage.
-v	Show tool version.

Methods for Sending MDIO Transactions

mlxmdio will attempt to send the MDIO transaction through a firmware interface if supported (on supported devices only). The mdio gateway values should be in the range of 0..10.

> **Note**
>
> Sending MDIO transactions via FW requires specification of the PCI device.

Example:

To read mlxmdio register, run the following command:

```
# mlxmdio -d /dev/mst/mt4123_pciconf0 -m 0x7:0x1 -a 0x0 -g 6
```

To write mlxmdio register, run the following command:

```
# mlxmdio -d /dev/mst/mt4123_pciconf0 -m 0x7:0x1 -a 0x0:0x0124 -g 6
```

To read block of 5 sequential operations through mlxmdio register, run the following command:

```
# mlxmdio -d /dev/mst/mt4123_pciconf0 -m 0x7:0x1 -a 0x14 -g 6 -r 5
0x0014:0x0010
0x0015:0x0003
0x0016:0x0030
0x0017:0x0040
0x0018:0x0050
```

Note

The address of the beginning of the block should be provided with the "-a" flag (e.g., `-a 0x14`).

To write block of operations through mlxmdio register, run the following command:

```
# mlxmdio -d /dev/mst/mt4123_pciconf0 -m 0x7:0x1 -g 6 -w input.txt
```

Note

Every line of the input file should be formatted as addr:data in hex.
The address of every line is not necessarily in a sequential order.

An example of the input file:

`0x0017:0x0040`

`0x0014:0x0010`

`0x0015:0x0003`

`0x0018:0x0050`

`0x0016:0x0030`

mlxreg Utility

The mlxreg utility allows users to obtain information regarding supported access registers, such as their fields and attributes. It also allows getting access to register data from firmware and setting access register data on firmware.

Registers can be get/set in unknown (RAW) mode by providing register ID and length.

Note

Unknown (RAW) mode is risky as no checks are performed, please consult with Support before using it.

mlxreg Usage

mst driver must be started prior to running mlxreg tool.

Some access registers depend on setup configuration such as link up/down. Invalid setup may cause failures.

To run mlxreg, use the following line:

```
mlxreg [options]
```

where:

-h |--help	Displays help message.
-v |--version	Displays version info.
-d |--device <device>	Performs operation for a specified mst device.
-a |--adb_file <adb_file>	An external ADB file
--reg_name <reg_name>	Known access register name
--reg_id <reg_ID>	Access register ID
--reg_len <reg_length>	Access register layout length (bytes)
-i |--indexes <idxs_vals>	Register indexes
-g |--get	Register access GET
-s |--set <reg_dataStr>	Register access SET
--show_reg <reg_name>	Prints the fields of a given reg access (must have reg_name)
--show_regs	Prints all available access registers
--yes	Non-interactive mode, answer yes to all questions
--i2c_secondary <i2c_secondary>	I^2C secondary address
-w |--overwrite	Set only specified fields, set unspecified fields to zero (only valid for SET command)
--full_path	Show field names with full hierarchical path. (Recommended to use it always to avoid field ambiguity)

Examples:

Show all available access registers (the example below shows a sample of the whole list):

```
mlxreg -d /dev/mst/mt4115_pciconf0 --show_regs
Available Access Registers
==============================================================================
CWTP
CWTPM
MCIA
MLCR
MPCNT
MPEIN
NCFG
PAOS
PDDR
PMDR
PMLP
PPAOS
PPCNT
PPLM
PPLR
PPRT
PPTT
PTAS
PTYS
ROCE_ACCL
SBCM
SBDCR
SBPM
SBPR
SBSR
SLRG
SLRP
SLTP
```

```
......
```

Query a single access register (PAOS):

```
mlxreg -d /dev/mst/mt4115_pciconf0 --show_reg PAOS
Field Name | Address (Bytes) | Offset (Bits) | Size (Bits) |
Access
====================================================================
oper_status  | 0x00000000 | 0  | 4 | RO
admin_status | 0x00000000 | 8  | 4 | RW
local_port   | 0x00000000 | 16 | 8 | INDEX
swid         | 0x00000000 | 24 | 8 | INDEX
e            | 0x00000004 | 0  | 2 | RW
ee           | 0x00000004 | 30 | 1 | WO
ase          | 0x00000004 | 31 | 1 | WO
====================================================================
```

Note: There might be indexes in access register fields that must be provided when setting or getting data.

Get access register data (PAOS with indexes: local port 1, swid 0):

```
mlxreg -d /dev/mst/mt4115_pciconf0 --reg_name PAOS --get --
indexes "local_port=0x1,swid=0x0"
Field Name | Data
=============================
oper_status  | 0x00000001
admin_status | 0x00000001
local_port   | 0x00000001
swid         | 0x00000000
e            | 0x00000000
ee           | 0x00000000
ase          | 0x00000000
```

```
==============================
```

Set access register data (PAOS with indexes: local_port 1 swid 0x0 and data: e 1):

```
mlxreg -d /dev/mst/mt4115_pciconf0 --reg_name PAOS --indexes
"local_port=0x1,swid=0x0" --yes --set "e=0x1"
You are about to send access register: PAOS with the following
data:
Field Name | Data
==============================
oper_status  | 0x00000002
admin_status | 0x00000001
local_port   | 0x00000001
swid         | 0x00000000
e            | 0x00000001
ee           | 0x00000000
ase          | 0x00000000
==============================
Do you want to continue ? (y/n) [n] : y
Sending access register...
```

Get access register data (PAOS (0x5006) in unknown mode (RAW) with indexes: local_port=0x1 swid=0x0):

```
mlxreg -d /dev/mst/mt4115_pciconf0 --reg_id 0x5006 --reg_len 0x10 -
-indexes "0x0.16:8=0x1,0x0.24:8=0x0" --get
Address    | Data
=======================
0x00000000 | 0x00010101
0x00000004 | 0x00000000
0x00000008 | 0x00000000
0x0000000c | 0x00000000
```

```
========================
```

Set access register data (PAOS in unknown mode (RAW) with indexes: local_port=0x1 swid=0x0 and data e 1):

```
mlxreg -d /dev/mst/mt4115_pciconf0 --reg_id 0x5006 --reg_len 0x10 --indexes "0x0.16:8=0x1,0x0.24:8=0x0" --yes --set "0x4.0:2=0x1"
You are about to send access register id: 0x5006 with the following data:
Address     | Data
========================
0x00000000 | 0x00010102
0x00000004 | 0x00000001
0x00000008 | 0x00000000
0x0000000c | 0x00000000
========================
Do you want to continue ? (y/n) [n] : y
Sending access register...
```

mlxlink Utility

The mlxlink tool is used to check and debug link status and related issues. The tool can be used on different links and cables (passive, active, transceiver and backplane).

Note

- In order for mlxlink to function properly, make sure to update the firmware version to the latest version.
- mlxlink is intended for advanced users with appropriate technical background.
- When using mlxlink to disable the port state ("--port_state dn" flag) on a NIC connected through a Socket-Direct connection,

the port must be disabled on both mst devices representing the physical port.

- Do not use mlxlink to disable the port connecting between the host and the unmanaged switch using ("--port_state dn") flag.
- When running mlxlink to show SLTP for 16nm technology, the "--advanced" flag should be added to the run command.
- mlxlink errors, warnings and notes are printed on stderr console.
- Setting the speeds (50GbE, 100GbE and 200GbE) for the new devices (NVIDIA ConnectX-6 and above, NVIDIA Quantum switches and above) requires specifying the number of lanes for the speed: mlxlink -d <dev> --speeds [50G_2X | 50G_1X | 100G_2X | 100G_4X | 200G_4X].
- In ConnectX-7 and later cards, configuring the loopback can be applied when the link is fully down (not in polling state).

mlxlink Usage

To run mlxlink:

```
mlxlink [OPTIONS]
```

where:

Options:

-h |--help	Display help message.
-v |--version	Display version info.
-d |--device <device>	Perform operation for a specified mst device
-p |--port <port_number>	Port Number
--port_type <port_type>	Port Type [NETWORK(Default)/PCIE/OOB]

--depth <depth>	Depth level of the DUT of some hierarchy (valid for PCIe port type only)
--pcie_index <pcie_index>	PCIe index number (Internal domain index) (valid for PCIe port type only)
--node <node>	The node within each depth (valid for PCIe port type only)
--json	Print the output in JSON format

Queries:

--show_links	Show valid PCIe links (valid for PCIe port type only)
-m |--show_module	Show Module Info
-c |--show_counters	Show Physical Counters and BER Info
-e |--show_eye	Show Eye Opening Info
--show_fec	Show FEC Capabilities
--show_serdes_tx	Show Transmitter Info
--show_tx_group_map <group_num>	Display all label ports mapped to group <group_num> (for NVIDIA Spectrum-2 and NVIDIA Quantum devices).
--show_device	General Device Info
--show_ber_monitor	Show BER Monitor Info. **Note:** The flag is not supported in HCAs.
--show_external_phy	Show External PHY Info **Note:** The flag is supported in NVIDIA Spectrum switch systems only.
--show_plr	Show Physical Layer Retransmission (PLR) Info **Note:** The flag is supported in NVL5 (and above) switch systems only.
--show_kr	Show Link Training Optimization (KR) Info **Note:** The flag is supported in NVL5 (and above) switch systems only.
--show_rx_recovery_counters	Show Rx Recovery Counters **Note:** The flag is supported in NVL5 (and above) switch systems only.
--show_peq	Show Link Periodic Equalization (PEQ) Info **Note:** The flag is supported in NVL5 (and above) switch systems only.

Commands:

-a |--port_state <port_state>		Configure Port State [UP(up)/DN(down)/TG(toggle)]
-s |--speeds <speeds>		Configure Speeds [speed1,speed2,...]
	--link_mode_force	Configure Link Mode Force (Disable AN)
-l |--loopback <loopback>		Configure Loopback Mode [NO(no loopback)/RM(phy re
-k |--fec <fec_override>		Configure FEC [AU(Auto)/NF(No-FEC)/FC(FireCode FEC)/
	--fec_speed <fec_speed>	Speed to Configure FEC [100G/50G/25G/...] (Default is A
--serdes_tx <params>		Configure Transmitter Parameters [polarity,ob_tap0,...]
	--serdes_tx_lane <transmitter_lane>	Transmitter Lane to Set (Optional - Default All Lanes)
	--database	Save Transmitter Configuration for Current Speed Perm
	--tx_params_override	Set the parameters according to Data Base only, otherw
--tx_group_map <group_num>		Map ports to group <group_num> (for NVIDIA Spectrun
	--ports <ports>	Ports to be mapped [1,2,3,4..]
--test_mode <prbs_mode>		Physical Test Mode Configuration [EN(enable)/DS(disabl
	--rx_prbs <rx_prbs_mode>	RX PRBS Mode [PRBS31(Default)/PRBS7/...] (Optional - [
	--tx_prbs <tx_prbs_mode>	TX PRBS Mode [PRBS31(Default)/PRBS7/...] (Optional - [
	--rx_rate <rx_lane_rate>	RX Lane Rate [EDR(Default)/25G/10G/...] (Optional - Defa
	--tx_rate <tx_lane_rate>	TX Lane Rate [EDR(Default)/25G/10G/...] (Optional - Defa
	--invert_tx_polarity	PRBS TX polarity inversion (Optional - Default No Inversi
	--invert_rx_polarity	PRBS RX polarity inversion (Optional - Default No Inversi
	--lanes	PRBS lanes to set (one or more lane separated by comm **Optional:** Default all lanes
	--rx_modulation	RX Modulation [NRZ/PAM4/PAM4_PRECODING/PAM4_N
	--tx_modulation	TX Modulation [NRZ/PAM4/PAM4_PRECODING/PAM4_N
-b |--ber_collect <csv_file>		Port Extended Information Collection [CSV File]
--amber_collect <csv_file>		AmBER Port Extended Information Collection For 16nm
--ber_limit <limit_criteria>		BER Limit Criteria [Nominal(Default)/Corner/Drift] (Optic
	--iteration <iteration>	Iteration Number of BER Collection

--pc	Clear Counters
--set_external_phy	Set External PHY **Note:** The flag is supported in NVIDIA Spectrum switch
--twisted_pair_force_mode <twisted_pair_force_mode>	Twisted Pair Force Mode [MA(Master)/SL(Slave)]
--cable	Perform operations on the cables
--dump	Dump cable pages in raw format
--ddm	Get cable Digital Diagnostic Monitoring information
--read	Perform read operation from specific page
--length <length>	Length of data to read in bytes (Optional - Default 1 byt
--page <pageNum>	Specific page number to read/write
--offset <offset>	Specific page offset to read/write
--write <bytes>	Perform write operation with specific data (list of bytes,
--prbs_select <side>	Module PRBS test mode side selector [MEDIA, HOST]
--prbs_mode <cmd>	Perform PRBS test mode on the Module [EN(Enable),DS
--generator_pattern <pattern>	Set PRBS generator pattern [PRBS31(default),PRBS23,P
--swap_generator	Enable PAM4 MSB <-> LSB generator swapping (Option
--invert_generator	Enable PRBS generator inversion (Optional)
--generator_lanes <lanes>	PRBS generator lanes to set (one or more lane separate
--checker_pattern <pattern>	Set PRBS checker pattern [PRBS31(default),PRBS23,PR
--swap_checker	Enable PAM4 MSB <-> LSB checker swapping (Optional)
--invert_checker	Enable PRBS checker inversion (Optional)
--checker_lanes <lanes>	PRBS checker lanes to set (one or more lane separated
--lane_rate <rate>	Set PRBS checker and generator lane rate [HDR(50G)(de
--show_diagnostic_info	Show PRBS diagnostic counters information
--clear_diagnostic_info	Clear PRBS diagnostic counters
--control_parameters	Show Module Control Parameters

--tx_equalization <value>	Set Module Tx Input Equalization in dB [NE(No Equalizat
--rx_emphasis <value>	Set Module RX Output Emphasis in dB. for CMIS, pre-er
--rx_post_emphasis <value>	Set Module Rx Post Emphasis in dB [NE(No Equalization
--rx_amplitude <value>	Set Module Rx Output Amplitude [0(100-400mV),1(300-
--margin	Read the SerDes eye margins per lane
--measure_time <time>	Measure time in seconds for single eye [10, 30, 60, 90, 1
--eye_select <eye_sel>	Eye selection for PAM4 [UP, MID, DOWN, ALL] (Default A
--lane <lane_index>	Run eye for specific lane index (Default all lanes)
--rx_error_injection	Enable the RX link deterioration
--mixer_offset0 <value>	Fine change to the center of the eye [0x0 to 0x7ff]
--mixer_offset1 <value>	Coarse change to the center of the eye [0x0 to 0x3ff]
--show_mixers_offset	Show mixer offset 0 and mixer offset 1
--rx_fec_histogram	Provide histogram of FEC errors. The result is divided to
--show_histogram	Show FEC errors histogram
--clear_histogram	Clears FEC errors histograms
--pcie_error_injection	Start/show PCIe error injection
--error_type <type>	PCIe error type [ABORT(0),BAD_DLLP_LCRC(1),BAD_TLP_LCRC(2),BAD_T
--error_duration <duration>	Error duration, depend on the error type
--injection_delay <delay>	Delay in micro-seconds before applying the error (Optio
--error_parameters <params>	Comma-separated parameters for selected error type (p
--dbdf <dbdf>	Destination bus device function, e.g af:00.0 (Optional us
--yes	Non-interactive mode, answer yes to all questions
--link_training <link_training_mode>	Link Training Optimization Configuration EN(enable)/DS(
--phy_recovery <phy_recovery_mode>	Configure PHY Recovery EN(enable)/DS(disable)
--recovery_type <type>	Configure Recovery Type host_serdes_feq/host_logic_re

--set_link_peq <interval>	Set link Periodic Equalization (PEQ) interval [10-40000] u Only multiples of 10 are supported.

Examples:

Get info of <device>, <port_number>:

```
mlxlink -d <device> -p <port_number>
```

Get info of <device>, <port_number> and BER Counters:

```
mlxlink -d <device> -p <port_number> -c
```

Get info of <device>, <port_number> and Transmitter Parameters:

```
mlxlink -d <device> -p <port_number> --show_serdes_tx
```

Configure Port State:

```
mlxlink -d <device> -p <port_number> --port_state UP
```

Configure Port Speeds:

```
mlxlink -d <device> -p <port_number> --speeds 25G,50G,100G
```

Configure FEC:

```
mlxlink -d <device> -p <port_number> --fec RS
```

Configure Port for Physical Test Mode:

```
mlxlink -d <device> -p <port_number> --test_mode EN (--rx_prbs
PRBS31 --rx_rate 25G --tx_prbs PRBS7 --tx_rate 10G --
invert_rx_polarity --invert_tx_polarity)
```

Perform PRBS Tuning:

```
mlxlink -d <device> -p <port_number> --test_mode TU
```

Note

RX and TX lane rates for new devices include the PAM4 speeds (50G_1X and 100G_2X).

eg:

```
mlxlink -d <device> --test_mode EN --rx_rate [normal
speeds | 50G_1X | 100G_2X] --tx_rate [normal speeds |
50G_1X | 100G_2X]
```

Note

The PRBS pattern configured in PAM4 rates is PRBSQ.

Cable operations:

```
mlxlink -d <device> --cable [Options]
```

Dump cable EEPROM pages:

```
mlxlink -d <device> --cable --dump
```

Get cable DDM information:

```
mlxlink -d <device> --cable --ddm
```

Read from cable:

```
mlxlink -d <device> --cable --read --page <page number> --offset
<bytes offset> --length <number of bytes>
```

Write to cable:

```
mlxlink -d <device> --cable --write <bytes separated by comma> --
page <page number> --offset <bytes offset>
```

Configure Transmitter Parameters (on lane, to database):

```
mlxlink -d <device> -p <port_number> --serdes_tx <polar- ity>,
<ob_tap0>,<ob_tap1>,<ob_tap2>,<ob_bias>,<ob_preemp_mode>,
<ob_reg>,<ob_leva> (--serdes_tx_lane <lane number>) (--database)
```

Configure Transmitter Parameters for 16nm devices:

```
mlxlink -d <device> -p <port_number> --serdes_tx <pre_2_tap>,
<pre_tap>,<main_tap>,<post_tap>,<ob_m2lp>,<ob_amp>
```

Getting PCIe links info:

```
mlxlink -d /dev/mst/mt41682_pciconf0 --port_type PCIE --
show_links
Valid PCIe Links
----------------
: depth, pcie_index, node, port
Link 1 : 3, 0, 0, 60
Link 2 : 3, 0, 1, 61
Link 3 : 3, 0, 2, 62
..
```

To query information for a specific link, the depth, pcie_index and node for the link must be specified:

```
mlxlink -d /dev/mst/mt41682_pciconf0 --port_type PCIE --depth 3 -
-pcie_index 0 --node 1 --show_serdes_tx --show_eye   PCIe
Operational (Enabled) Info
-------------------------------
Depth, pcie index, node : 3, 0, 1
Link Speed Active (Enabled) : 8G-Gen 3 (16G-Gen 4)
Link Width Active (Enabled) : 2X (16X)

EYE Opening Info (PCIe)
-----------------------
Physical Grade : 84, 84
Height Eye Opening [mV] : 1194, 1194
```

```
Phase Eye Opening [psec] : 84, 84

Serdes Tuning Transmitter Info (PCIe)
-------------------------------------
Serdes TX parameters              : Pol ,tap0 ,tap1 ,tap2 ,bias
,preemp_mode ,reg ,leva
Lane 0                            : 0    ,21   ,92   ,7    ,15   ,1
,10   ,9
Lane 1                            : 1    ,21   ,92   ,7    ,15   ,1
,10   ,9
Lane 2                            : 0    ,21   ,92   ,7    ,15   ,1
,10   ,9
Lane 3                            : 1    ,21   ,92   ,7    ,15   ,1
,10   ,9
Lane 4                            : 0    ,21   ,92   ,7    ,15   ,1
,10   ,9
Lane 5                            : 1    ,21   ,92   ,7    ,15   ,1
,10   ,9
Lane 6                            : 0    ,21   ,92   ,7    ,15   ,1
,10   ,9
Lane 7                            : 1    ,21   ,92   ,7    ,15   ,1
,10   ,9
```

To print the output in JSON format:

```
mlxlink -d <device> --show_module --json
```

To show ports group map (for NVIDIA Quantum and NVIDIA Spectrum-2):

```
mlxlink -d<device> --show_tx_group_map 0
```

To assign ports to a specific group on NVIDIA Quantum and NVIDIA Spectrum-2:

```
mlxlink -d <device> --tx_group_map 1 -ports 1,2,3,5,4,8,7,8,9,10,11
```

To show histogram of FEC errors:

```
mlxlink -d /dev/mst/mt4125_pciconf0 --rx_fec_histogram --
show_histogram
```

To clear histogram:

```
mlxlink -d /dev/mst/mt4125_pciconf0 --rx_fec_histogram --
clear_histogram
```

Margin Scan Tool

The margin scan tool is used for scanning PCIe or Network ports (in EDR\25G or HDR\PAM4 speeds).

> **Note**
>
> If the margin scan fails with an "eye scan not completed" message, perform a reboot and run the scan again.

To enable the margin scan with a measure time of 10 seconds:

```
mlxlink -d <device> --port_type PCIE -margin -measure_time 10
```

To enable the margin scan for Multi-host or Socket Direct systems through:

- depth, pcie_index and node:

```
mlxlink -d <device> --port_type PCIE -depth 0 -pcie_index 1 -
node 0 -margin -measure_time 30
```

- The local port (can be shown by the `-show_links` command):

```
mlxlink -d <device> --port_type PCIE -port 1 -margin -
measure_time 10
```

RX Error Injection

Allows modifying the Eye Center capability by changing the mixer_offset0 (fine change) and mixer_offset1 (coarse change) flags for 28nm products to produce RX errors.

Flags Usage

- To change the mixers values:

```
mlxlink -d /dev/mst/mt4117_pciconf0 --rx_error_injection --
mixer_offset0 0x200 --mixer_offset1 0x305
```

> **Note**
>
> Modifying `mixer_offset0` and `mixer_offset1` flags can change the Eye Center and might cause link degradation.

- To query the mixers values:

```
 mlxlink -d /dev/mst/mt4117_pciconf0 --rx_erro r_injection --
show_mixers_offset
```

Rx-to-Tx Loopback Mode Activation

This capability enables Rx-to-Tx remote loopback mode.

Prerequisite

- The port should be disabled and configured with Force mode [Disable the Auto-negotiation]
- Remote loopback is supported for 25G/50G per lane speed
- If the NIC has 2 ports, both ports should be configured with the same speed

To enable Rx-to-Tx remote loopback mode:

1. Disable the port.

```
mlxlink -d /dev/mst/mt4123_pciconf0 –port_state DN
```

2. Configure the link mode to Force [Disable the Auto-negotiation].

```
mlxlink -d /dev/mst/mt4123_pciconf0 --speeds 100G --
link_mode_force
```

3. Configure loopback with remote loopback [RM].

```
mlxlink -d /dev/mst/mt4123_pciconf0 –loopback RM
```

4. Enable the port.

```
mlxlink -d /dev/mst/mt4123_pciconf0 —port_state UP
```

To return to the normal link operation:

1. Disable the port.

```
mlxlink -d /dev/mst/mt4123_pciconf0 —port_state DN
```

2. Clear the loopback configuration using the "NO loopback" option.

```
mlxlink -d /dev/mst/mt4123_pciconf0 —loopback NO
```

3. Enable the port.

```
mlxlink -d /dev/mst/mt4123_pciconf0 —port_state UP
```

Module PRBS Test Mode

The module PRBS test mode can be performed by using the new flags under the `--cable` command.

Note

- This feature supports Active/Optical CMIS modules only.
- Either the media or host side can run with PRBS mode.

- To enable the PRBS test mode, the module should be plugged in and active.

Enabling\Disabling The Module PRBS Test Mode

To enable the module PRBS test mode, the side of the module should be selected using the `--prbs_select` flag. After providing the `--cable` flag, either the HOST or the MEDIA side should be selected, so the `--prbs_mode` <EN\DS> can be used to enable or disable the PRBS test mode process.

For example, the following command will enable the PRBS test mode on the HOST side of the module:

```
mlxlink -d /dev/mst/mt53104_pciconf0 --port 3 --cable --
prbs_select HOST --prbs_mode EN
```

The command above will put the HOST side of the module in PRBS test mode with default Checker and Generator parameters.

The Checker and Generator parameters can be overridden while enabling the PRBS test mode according to their related flags in the help menu:

```
mlxlink -d /dev/mst/mt53104_pciconf0 --port 3 --cable --
prbs_select HOST --prbs_mode EN --checker_pattern PRBS13 --
invert_checker --generator_pattern PRBS31 --swap_generator --
lane_rate HDR
```

To disable the PRBS test mode, the following command can be executed:

```
mlxlink -d /dev/mst/mt53104_pciconf0 --port 3 --cable --
prbs_select HOST --prbs_mode DS
```

Getting Module Information

Some of the module information can be presented using the `--show_module` flag.

For example, the following command will present the module information for the module connected to port 1 on the switch:

```
mlxlink -d /dev/mst/mt53120_pci_cr0 --port 1 --show_module

Module Info
-----------
Identifier                           : OSFP
Compliance                           : IB HDR,200GBASE-SR4
Cable Technology                     : 850 nm VCSEL
Cable Type                           : Optical Module (separated)
OUI                                  : Nvidia
Vendor Name                          : NVIDIA
Vendor Part Number                   : MMA4U00-HS
Vendor Serial Number                 : MT2134FT06348
Rev                                  : A3
Wavelength [nm]                      : 850
Transfer Distance [m]                : 10
Attenuation (5g,7g,12g,25g,53g)[dB]: N/A
FW Version                           : 40.55.0
Digital Diagnostic Monitoring        : Yes
Power Class                          : 12.0 W max
CDR RX                               : ON,ON,ON,ON,ON,ON,ON,ON
CDR TX                               : ON,ON,ON,ON,ON,ON,ON,ON
LOS Alarm                            : N/A
Temperature [C]                      : 36 [-10..80]
Voltage [mV]                         : 3280 [3100..3500]
Bias Current [mA]                    :
7.460,7.490,7.292,7.348,7.430,7.200,7.392,7.300 [5.492..8.5]
Rx Power Current [dBm]               : -40,-40,-40,-40,-40,-40,-40,-40
[-13.372..5.4]
```

```
Tx Power Current [dBm]                  :
0.473,0.477,0.282,0.434,0.766,0.645,0.500,0.500 [-11.427..5.4]
Intra-ASIC Latency [ns]                 : N/A
Module Datapath Latency [ns]            : N/A
Round Trip Latency [ns]                 : N/A
SNR Media Lanes [dB]                    : N/A
SNR Host Lanes [dB]                     : N/A
IB Cable Width                          : N/A
Memory Map Revision                     : 64
Linear Direct Drive                     : 0
Cable Breakout                          : Unspecified
SMF Length                              : N/A
MAX Power                               : 48
Cable Rx AMP                            : 1
Cable Rx Emphasis (Pre)                 : 0
Cable Rx Post Emphasis                  : 0
Cable Tx Equalization                   : 0
Wavelength Tolerance                    : 15.0nm
Module State                            : Ready state
DataPath state [per lane]               :
DPActivated,DPActivated,DPActivated,DPActivated,DPActivated,DPActi
Rx Output Valid [per lane]              : 0,0,0,0,0,0,0,0
Nominal bit rate                        : N/A
Rx Power Type                           : Average power
Manufacturing Date                      : 24_08_21
Active Set Host Compliance Code         : IB HDR
Active Set Media Compliance Code        : 200GBASE-SR4
Error Code Response                     : ConfigUndefined
Module FW Fault                         : 0
DataPath FW Fault                       : 0
Tx Fault [per lane]                     : 0,0,0,0,0,0,0,0
Tx LOS [per lane]                       : 0,0,0,0,0,0,0,0
Tx CDR LOL [per lane]                   : 0,0,0,0,0,0,0,0
Rx LOS [per lane]                       : 0,0,0,0,0,0,0,0
Rx CDR LOL [per lane]                   : 1,1,1,1,1,1,1,1
Tx Adaptive EQ Fault [per lane]         : 0,0,0,0,0,0,0,0
```

```
mlxlink -d /dev/mst/mt53104_pciconf0 --port 3 --cable --
prbs_select HOST --clear_diagnostic_info
```

PRBS Diagnostic Counters Information

After performing the PRBS test mode, the module counters can be queried by using the following command:

```
mlxlink -d /dev/mst/mt53104_pciconf0 --port 3 --cable --
prbs_select HOST --show_diagnostic_info
```

The module PRBS test mode counters can be cleared by using the following command, which will clear the diagnostic counters on the HOST side only:

```
mlxlink -d /dev/mst/mt53104_pciconf0 --port 3 --cable --
prbs_select HOST --clear_diagnostic_info
```

Module Control Parameters

Some of the module parameters can be controlled by mlxlink after providing the `--control_paramenter` flag, which can be executed under the `--cable` flag.

The possible parameters can be controlled as follows:

- Reading and configuring Tx Equalization
- Reading and configuring Rx Emphasis (PreCursor & PostCursor)
- Reading and configuring Rx Amplitude

> **Note**
>
> - To apply the changes, the link should be disabled first.
> - After configuring a new parameter, the link should be raised again to allow the firmware to load the new configuration.
> - Cable control parameters are valid for active\optical modules only.

Querying and Configuring The Module Control Parameters

To query the currently configured module control parameters, the `--control_parameters` flag can be used under the `--cable` flag as follows:

```
mlxlink -d /dev/mst/mt53104_pciconf0 --cable --control_parameters
...
Module Control Parameters
-------------------------
TX Equalization                 : 1dB
RX Emphasis (pre)               : 2.5dB
RX Post Emphasis                : No Equalization
RX Amplitude                    : 600-1200 mV (P-P)
```

To configure the module control parameters, the following command can be executed:

```
mlxlink -d /dev/mst/mt53104_pciconf0 --cable --control_parameters
--tx_equalization 2 --rx_amplitude 1
```

Tool Usage with NIC vs. Switch (-p Flag)

When using the mlxlink tool with an adapter card, notice that the "label_port" `-p` flag should not be used. To address different ports, please use different MST devices.

For example:

To address port 1 when using ConnectX-4:

```
mlxlink -d /dev/mst/mt4115_pciconf0
```

To address port 2, use:

```
mlxlink -d /dev/mst/mt4115_pciconf0.1
```

Note

- Any mlxlink command for a switch should include the "`-p`" flag to address the specific port in the switch.
- When working with an adapter card, if an MTUSB is used for communication with the NVIDIA NIC, to address port 2, use `mlxlink -d /dev/mst/mt4115_pciconf0 --gvmi_address <0xAddress>`.

Tool Usage on NVIDIA Quantum HDR Switch Systems with Split Ports

If the split port number is not provided by the ibdiagnet tool, to use mlxlink on NVIDIA Quantum HDR based switch systems split ports, run:

```
mlxlink -d lid-<LID> -p <formula>
```

Formula:

In case of 2X port:

- 1- port_num = round_down[(Iblinkinfo_port_num + 1)*0.5]
- 2- if (Iblinkinfo_port_num + 1) modulo 2 =1 then append '/2' to port_num

In case of 4X port, use only item #1 above.

Example:

```
43 23[ ] ==( 2X 53.125 Gbps Active/ LinkUp)==> mlxlink -d lid-43 -p
12
43 24[ ] ==( 2X 53.125 Gbps Active/ LinkUp)==> mlxlink -d lid-43 -p
12/2
```

Tool Usage on NVIDIA Quantum-2 NDR Switch Systems

In NVIDIA Quantum-2 NDR switch systems, there are 32-OSFP cages (8x), where each one holds 2 (4x) ports instead of 1, and each port can be accessed by providing the cage number and the port in the cage - "Cage/Port".

```
mlxlink -d <mst deivce> -p <Cage>/<Port>
```

If the split profile is ready, it is possible to access the split ports by providing the number of split to the port flag, e.g.:

- To access the main port of 15/2:

```
mlxlink -d <mst deivce> -p 15/2
```

- To access the split port of 15/2:

```
mlxlink -d <mst deivce> -p 15/2/2
```

PCIe

Link Speed and Width

For PCIe link speed and width, use the following flag: `--port_type PCIE`.

```
PCIe Operational (Enabled) Info
-------------------------------
Depth, pcie index, node     : [Depth, pcie index, node]
Link Speed Active (Enabled) : [Freq - Gen]
Link Width Active (Enabled) : [Width]
```

PCIe Switch

When using NVIDIA ConnectX-5 and newer devices, the PCIe interface can be configured for a PCIe switch. When the PCIe switch is enabled, the depth, pcie_index and node parameters are needed in order to specify the PCIe port from which the requested information (such as counters or eye info) is gathered.

Parameters	Description
Depth	This parameter defines the number of layers from the Root Complex to the specific port. • For NVIDIA ConnectX adapter cards multi-host mode, the depth should be set to 0.

Parameters	Description
	• For NVIDIA BlueField/BlueField-2 JBoF, the depth should be set to 3.
Pcie_index	This parameter defines the root complex ID or host index. • For NVIDIA ConnectX adapter cards multi-host mode, the pcie_index is the host index (0–3). • For NVIDIA BlueField/BlueField-2 JBoF, the pcie_index is always 0.
Node	This parameter defines the specific PCIe port. • For NVIDIA ConnectX adapter cards multi-host mode, the node is always 0 for each host_index. • For NVIDIA BlueField JBoF mode, this parameter range is 0x0–0xF, which amounts for up to 16 possible ports for BlueField JBoF. • For NVIDIA BlueField-2, this parameter's range is 0x0–0x7. **Note:** For NVIDIA BlueField/BlueField-2 SmartNIC mode, the PCIe link information can only be gathered from the external host. The PCIe interface status cannot be retrieved from the Arm side. When retrieving the PCIe link information from the external host, there is no need to specify the depth, pcie_index and node.

Example: NVIDIA BlueField JBoF Mode

```
# mlxlink -d /dev/mst/mt41682_pciconf0 --port_type pcie --depth 3
--pcie_index 0 --node 4 -c



PCIe Operational (Enabled) Info
-------------------------------
Depth, pcie index, node          : 3, 0, 4
Link Speed Active (Enabled)      : 8G-Gen 3 (16G-Gen 4)
Link Width Active (Enabled)      : 2X (2X)

Management PCIe Timers Counters Info
------------------------------------
dl down                          : 0
```

```
Management PCIe Performance Counters Info
-----------------------------------------
RX Errors                       : 0
TX Errors                       : 0
CRC Error dllp                  : 0
CRC Error tlp                   : 0
```

Link Counters

For PCIe counters information, use the `--port_type PCIE -c` flag.

```

Management PCIe Timers Counters Info
------------------------------------
dl down                         : [link down counter]

Management PCIe Performance Counters Info
-----------------------------------------
RX Errors                       : [Rx Errors]
TX Errors                       : [Tx Errors]
CRC Error dllp                  : [CRC Errors dllp]
CRC Error tlp                   : [CRC Errors tlp]
```

- RX Errors: indicate the number of transitions to recovery required due to framing errors and CRC (dlp and tlp) errors.
- TX Errors: indicate the number of transitions to recovery required due to EIEOS and TS errors.
- CRC Error dllp: indicate CRC error in Data Link Layer Packets.
- CRC Error tlp: indicate CRC error in Transaction Layer Packet.

Example:

```
# mlxlink -d /dev/mst/mt4123_pciconf0 --port_type PCIE -c

PCIe Operational (Enabled) Info
-------------------------------
Depth, pcie index, node          : 0, 0, 0
Link Speed Active (Enabled)      : 16G-Gen 4 (16G-Gen 4)
Link Width Active (Enabled)      : 16X (16X)

Management PCIe Timers Counters Info
------------------------------------
dl down                          : 3

Management PCIe Performance Counters Info
-----------------------------------------
RX Errors                        : 0
TX Errors                        : 16
CRC Error dllp                   : 0
CRC Error tlp                    : 0
```

Link Eye Opening and Grade

For PCIe link physical grade and eye opening information, use the `--port_type PCIE -e` flag.

```
EYE Opening Info (PCIE)
------------------------
Physical Grade : [Grade0, Grade1, Grade2, Grade3, Grade4, Grade5,
Grade6, Grade7, Grade8, Grade9, Grade10, Grade11, Grade12,
Grade13, Grade14, Grade15]
Height Eye Opening [mV] : [Height0, Height1, Height2, Height3,
Height4, Height5, Height6, Height7, Height8, Height9, Height10,
Height11, Height12, Height13, Height14, Height15]
```

```
Phase Eye Opening [psec] : [Phase0, Phase1, Phase2, Phase3,
Phase4, Phase5, Phase6, Phase7, Phase8, Phase9, Phase10, Phase11,
Phase12, Phase13, Phase14, Phase15]
```

Example:

```
# mlxlink -d /dev/mst/mt4123_pciconf0 --port_type PCIE -e

PCIe Operational (Enabled) Info
-------------------------------
Depth, pcie index, node          : 0, 0, 0
Link Speed Active (Enabled)      : 16G-Gen 4 (16G-Gen 4)
Link Width Active (Enabled)      : 16X (16X)

EYE Opening Info (PCIe)
-----------------------
Physical Grade                   :  57279, 56340, 59340, 61824, 55140,
60501, 61530, 57392, 61573, 58930, 62752, 60421, 57188, 59796, 60066, 60847
Height Eye Opening [mV]          :    292,   288,   314,   325,
278,   310,   319,   299,   316,   318,   343,   323,   310,   311,
335,   318
Phase  Eye Opening [psec]        :     30,    30,    30,    30,
30,    30,    30,    30,    30,    28,    28,    28,    28,    30,
28,    30
```

Pass/Fail Criteria

SLRED (ConnectX-6/ConnectX-6 Dx/ConnectX-6 Lx)

```
mlxlink -d [device] --port_type PCIE --margin
```

Gen3

Gen3	
Eye Grade	Figure of Merit (FOM)
0 < Eye Grade < 700	FAIL
700 < Eye Grade < 2300	Gray area
2300 < Eye Grade	PASS

Gen4

Gen4	
Eye Margin	FOM
0 < Eye Grade < 150	FAIL
150 < Eye Grade < 400	Gray area
400 < Eye Grade	PASS

PCIE Error Injection

This test feature allows errors injection over the PCI links. It is used to verify that the system can handle the PCIe errors, which rarely occur in regular usage.

The ConnectX-7 device includes testability features that can be configured to act as an error injection 'exerciser' in order to test other components in the system. This is supported when the ConnectX-7 is used as a PCIe switch.

> **Note**
>
> - This is a PCIe related feature that should be run over PCIe links only (`--port_type PCIE`) with specific depth, PCIe index and node (DPN).
> - If the DPN is not provided, the tool will take the default values - 0,0 and 0, respectively.

- The mapping between the BDF and its DPN can be found by executing the `show_links` command (see example below).

Error Types

ID	Error Type	Description	Unit	Additional Parameters (`--errors_parameters`)
0	ABORT	Cancels the current pending error, if exists.	NA	NA
1	BAD_DLLP_LCRC	Flips a bit in the LCRC of the next " `error_duration` " DLLPs that are transmitted through the port.	Packets	NA
2	BAD_TLP_LCRC	Flips a bit in the LCRC of the next " `error_duration` " TLPs that are transmitted through the port. The packets are VDM TLPs that are sent by the port to the destination BDF - " `dbdf` ".	Packets	NA
3	BAD_TLP_ECRC	Flips a bit in the ECRC of the next "**error_duration**" TLPs that are transmitted through the port. The packets are VDM TLPs that are sent by the port to the destination BDF - "**dbdf**".	Packets	NA
4	ERR_MSG	Sends an error signaling message to the RC.	Packets	Parameter 0: message type 0 - Correctable

				1 - Nonfatal 2 - Fatal
5	MALFORMED_ TLP	Sends an “**error_duration**” PM_ACTIVE_STATE_NACK message to the destination BDF - “**dbdf**” with TC=1 instead of 0.	Packets	NA
6	POISONED_TLP	Sends an “**error_duration**” VDMs with data to the destination BDF - “**dbdf**” with EP = 1.	Packets	NA
7	UNEXPECTED_CPL	Sends “**error_duration**” completions to the destination BDF - “**dbdf**” with 0xff tag.	Packets	NA
8	ACS_VIOLATION	Sends “**error_duration**” VDMs to the destination BDF - “**dbdf**” with source_bdf=0.	Packets	NA
100	SURPRISE_LINK_DOWN	Sets a port state to DETECT.	NA	NA
101	RECEIVER_ERROR	Sends a clock instead of data for “**error_duration**” usecs. A value of 0 in ‘**error_duration’** means that this error must be toggled by the firmware as fast as possible.	uSec	NA

PCIe Error Injection Inputs

The following values should be provided in the error injection command line. Some values may be optional according to the error type.

Input	Command Line Flag	Description	Obligatory	Default
Error Type	`--error_type`	Error type according to the table above.	Yes	-
Error Duration	`--duration`	The minimal number of packets with this error that will be sent, or the minimal amount of time that this error state would be applied.	No	1
Injection Delay	`--injection_delay`	Delay in microseconds before the error is applied. This allows time for the completion to return to the tool caller correctly. A higher value can be used to allow the system to get to a lower power state.	No	0
Destination BDF	`--dbdf`	Destination BDF. Relevant for some of the errors that require packet generation. See error table above.	No	0:00.0
Additional Parameters	`--errors_parameters`	Additional parameters according to the error type. See error table above.	No	0

Mlxlink will trigger the firmware to start the error injection process by providing the `--pcie_error_injection` flag with the requested configuration parameters.

Note that the command returns immediately, but the error injection can take longer to complete (according to the error duration and injection delay inputs).

When the tool is run without the parameters above, it will query the error injection state – Whether it is ready to start a new error injection, or it is in the middle of the previous injection.

Usage Example

Start the process by performing error injection with error type UNEXPECTED_CPL.

This example shows how to start the error injection by sending 5 unexpected completion packets. The packets (of error type UNEXPECTED_CPL (id 7)) are directed from BDF 05:00.0 to BDF 06:0.0 after 500µs of sending the command in the following environment:

PCIe Component	BDF
Root port	00:01.0
(exerciser) PCIe Switch Upstream port	01:00.0
(exerciser) PCIe Switch Downstream port	05:00.0
Endpoint	06:00.0

To get the related depth, pcie_index and node for the specific BDF 05:00.0, the `show_links` command should be executed as follows:

```
mlxlink -d /dev/mst/mt4129_pciconf0 --port_type PCIE --show_links

Valid PCIe Links
----------------
Legend                          : depth ,pcie_index ,node ,port
,bdf
Link 1                          : 0     ,0          ,0    ,0
,01:00.0
Link 2                          : 3     ,0          ,0    ,60
,05:00.0
```

In this case, the depth, pcie_index, and node flags for the downstream port should be 3, 0, and 0, respectively.

Then, the following command can be executed to start the process:

```
mlxlink -d /dev/mst/mt4129_pciconf0 --port_type PCIE --depth 3 --
pcie_index 0 --node 0 --pcie_error_injection --error_type
```

```
UNEXPECTED_CPL --error_duration 5 --dbdf 06:00.0 --
injection_delay 500

PCIe Operational (Enabled) Info
-------------------------------
Depth, pcie index, node          : 3, 0, 0
Link Speed Active (Enabled)      : 16G-Gen 4 (16G-Gen 4)
Link Width Active (Enabled)      : 16X (16X)

PCIe error injection might cause a PCIe bus failures or a system
hang
Do you want to continue? yes
Starting PCIe Error Injection...
```

Query Error Injection Status

After sending the configuration command, the progress of the process can be checked by executing the tool with the `pcie_error_injection` flag only:

```
mlxlink -d /dev/mst/mt4129_pciconf0 --port_type PCIE --depth 3 --
pcie_index 0 --node 0 --pcie_error_injection

PCIe Operational (Enabled) Info
-------------------------------
Depth, pcie index, node          : 3, 0, 0
Link Speed Active (Enabled)      : 16G-Gen 4 (16G-Gen 4)
Link Width Active (Enabled)      : 16X (16X)

PCIe Error Injection Info
-------------------------
Error Injection Status                  : In progress
Error Injection Type                    : UNEXPECTED_CPL
```

```
Error Injection Duration                    : 5 Packets
```

Once the process is complete, the output will be changed to "ready". This means that another error injection request can be submitted:

```
mlxlink -d /dev/mst/mt4129_pciconf0 --port_type PCIE --depth 3 --
pcie_index 0 --node 0 --pcie_error_injection

PCIe Operational (Enabled) Info
-------------------------------
Depth, pcie index, node         : 3, 0, 0
Link Speed Active (Enabled)     : 16G-Gen 4 (16G-Gen 4)
Link Width Active (Enabled)     : 16X (16X)

PCIe Error Injection Info
-------------------------
Error Injection Status                      : Ready
 Error Injection Type                       : N/A
Error Injection Duration                    : N/A
```

mlxfwstress Utility

Note

The tool can support new devices only once the tool is upgraded to its latest version.

mlxfwstress enables/disables various firmware stress flows. It can work in multiple modes:

- Enable/disable a specific set of stress types
- Clear all stress types
- Random mode:
    - Single mode - choose one stress type in each iteration and enable/disable it
    - Wild-mode- choose multiple stress types in each iteration and enable/disable them

  Each time a stress type is chosen in a random iteration, the opposite operation is done on it (e.g., if a stress type is turned on, in the next iteration it will be turned off and vice versa).

- Toggle mode:
    - Turns on and off the list of stress types alternating. Can be used with iterations.

      > **Note**
      >
      > To disable a stressor while in toggling mode, first you must disable the mlxfwstress tool, and only after that disable the stressor.

- Clear semaphore:

  Note: This functionality is supported in ConnectX-3 Pro adapter cards only.

mlxfwstress Synopsis

```
# mlxfwstress [-d|--dev <DeviceName>] [-h|--help] [-v|--version]
[-o|--operation <Operation>] [--rand-mode <Random mode>] [-t|--
stress-type <Stress type>] [--iterations <Iterations>] [--stress-
```

```
delay <Stress delay>] [--max-rand-on <Max rand on>] [--hang-type
<Hang type>] [--seed <seed>] [--toggle-time <x,y>]
```

where:

-d|--dev <DeviceName>	Perform operation for a specified device
-h|--help	Show this message and exit
-v|--version	Show the executable version and exit
-o|--operation <Operation>	Choose operation: on, off, clear_all, random query, clear_semaphore
--rand-mode <Random mode>	Choose a random mode: single, wild
-t|--stress-type <Stress type>	Specify a list of stress types separated by comma. (*See Stress Types.)*
--iterations <Iterations>	Specify the number of iterations.
--stress-delay <Stress delay>	Specify the stress delay in seconds (can be float). Note: Some stress flows may take more time. Recommended values: 0-1
--max-rand-on <Max rand on>	Specify the maximal time a stress is allowed to be on in random mode in seconds. Recommended values (0,1] Default is 1
--hang-type <Hang type>	Specify a list of hang types separated by comma. (*See Hang Types.)*
--seed <seed>	Specify the seed for the random.
--toggle-time <x,y>	Toggle time after off, both in seconds (can be float). If y is not supplied the tool will use equal values for x and y

Stress Types

ConnectX-4/ConnectX-4 Lx/ConnectX-5 Adapter Cards Stress Types

The following are the stress types available for ConnectX-4/ConnectX-4 Lx/ConnectX-5 adapter cards:

Category	Stress Type	Description	Notes
Transparent	PAUSE_STORM_GENERATION	Generates pause frames from the device toward the network	
	INVALIDATE_INTERNAL_CACHE_RX_1	Invalidates STE cache	
	INVALIDATE_INTERNAL_CACHE_RX_2	Invalidates qp L0 cache (RX)	
	INVALIDATE_INTERNAL_CACHE_RX_3	Invalidates dct L0 cache (RX)	
	INVALIDATE_INTERNAL_CACHE_RX_4	Invalidates scatter list cache in RX	
	INVALIDATE_INTERNAL_CACHE_CQ	Invalidates CQC cache	
	INVALIDATE_INTERNAL_CACHE_SX1	Invalidates SXDC cache	
	INVALIDATE_INTERNAL_CACHE_RX_5	Invalidates LDB cache	
	INVALIDATE_INTERNAL_CACHE_GENERAL_1	Invalidates RO caches	
	INVALIDATE_INTERNAL_CACHE_SX2	Invalidates pkey cache (SX)	
	INVALIDATE_INTERNAL_CACHE_SX3	Invalidates guid cache (SX)	

Category	Stress Type	Description	Notes
	INVALIDATE_INTERNAL_CACHE_QP	Invalidates QPC (main QP cache unit)	
Hang FW/HW	PACKET_DROP	Drops N packets on portx	This type require following extra fl • num_of_pa - 8 bit (max • port_num - (should be 2)

ConnectX-3 Pro Adapter Cards Stress Types

The following are the stress types available for ConnectX-3 Pro adapter cards:

> **Note**
>
> Stressors in "Transparent" category that are active for more than 100 msec, may cause resiliency.

Category	Stress Type	Description
Transparent	STOP_CE_INSTAGE_EQE	Stops sending EQEs created by the hardware (not the ones created by the firmware).
	STOP_EDBH	Stops the handling of external doorbells.
	STOP_IDBH	Stops the handling of internal doorbells.
	STOP_QPC_MISS_MACHINE_0 STOP_QPC_MISS_MACHINE_1 STOP_QPC_MISS_MACHINE_2 STOP_QPC_MISS_MACHINE_3	Spots reading a QPC from the ICM on a miss-blocking hardware/firmware that accesses the QPC

Category	Stress Type	Description
	LOCK_CEGW	Locks the CQE gateway.
	LOCK_OBGW_TPT LOCK_OBGW_TCU LOCK_OBGW_SXD	Locks the OBGW (access to the host memory gateway).
	LOCK_QPCGW_RX	Locks QPCGW.
	LOCK_SEMAPHORE_IPC_RX0 LOCK_SEMAPHORE_IPC_RX1 LOCK_SEMAPHORE_IPC_LDB LOCK_SEMAPHORE_IPC_SX1	Locks the IPC semaphore.
	INVALIDATE_CACHES	Invalidates caches.
Performance	STOP_SXP_VL_ARB_PORT1 STOP_SXP_VL_ARB_PORT2	Stops transmission of packets to the wire. Causes head-of-line packet drop (HLL) if enabled.
	RX_BACKPRESSURE	Stops the RX pipe - back-pressure to wire- sending tx pauses.
	DROP_PACKETS_TX	Drops packets on the TX side.

Turning On Stress Types

To turn on a specific stress type:

```
mlxfwstress -d mt4103_pciconf0 -o on -t STOP_CE_INSTAGE_EQE
-------------------------------------------------
Operation: [ON]
-------------------------------------------------
Turning ON stress type: stop_ce_instage_eqe -PASSED
```

To turn on a set of stress types:

```
mlxfwstress -d mt4103_pciconf0 -o on -t
STOP_CE_INSTAGE_EQE,STOP_QPC_MISS_MACHINE_3,LOCK_SEMAPHORE_IPC_RX1
```

```
--------------------------------------------------
Operation: [ON]
--------------------------------------------------
Turning ON stress type: stop_ce_instage_eqe -PASSED
Turning ON stress type: stop_qpc_miss_machine_3 -PASSED
Turning ON stress type: lock_semaphore_ipc_rx1 -PASSED
```

To turn on all the available stress types:

```
mlxfwstress -d mt4119_pciconf0 -t ALL -o on
Random seed: [1587969653]
--------------------------------------------------
Operation:         [ON]
--------------------------------------------------
Turning ON stress type: INVALIDATE_INTERNAL_CACHE_CQ
-PASSED
Turning ON stress type: INVALIDATE_INTERNAL_CACHE_GENERAL_1
-PASSED
Turning ON stress type: INVALIDATE_INTERNAL_CACHE_QP
-PASSED
Turning ON stress type: INVALIDATE_INTERNAL_CACHE_RX_1
-PASSED
Turning ON stress type: INVALIDATE_INTERNAL_CACHE_RX_2
-PASSED
Turning ON stress type: INVALIDATE_INTERNAL_CACHE_RX_3
-PASSED
Turning ON stress type: INVALIDATE_INTERNAL_CACHE_RX_4
-PASSED
Turning ON stress type: INVALIDATE_INTERNAL_CACHE_RX_5
-PASSED
Turning ON stress type: INVALIDATE_INTERNAL_CACHE_SX1
-PASSED
Turning ON stress type: INVALIDATE_INTERNAL_CACHE_SX2
-PASSED
```

```
Turning ON stress type: INVALIDATE_INTERNAL_CACHE_SX3
-PASSED
```

Turning Off Stress Types

To turn off a specific stress type:

```
mlxfwstress -d mt4103_pciconf0 -o off -t STOP_CE_INSTAGE_EQE
-------------------------------------------------
Operation: [OFF]
-------------------------------------------------
Turning OFF stress type: stop_ce_instage_eqe -PASSED
```

To turn off a set of stress types:

```
mlxfwstress -d mt4103_pciconf0 -o off -t
STOP_CE_INSTAGE_EQE,STOP_QPC_MISS_MACHINE_3,LOCK_SEMAPHORE_IPC_RX1
-------------------------------------------------
Operation: [OFF]
-------------------------------------------------
Turning OFF stress type: stop_ce_instage_eqe -PASSED
Turning OFF stress type: stop_qpc_miss_machine_3 -PASSED
Turning OFF stress type: lock_semaphore_ipc_rx1 -PASSED
```

Querying the Stress Types

To query the state of all stress types:

```
mlxfwstress -d mt4117_pciconf0 -o query -t ALL
```

```
--------------------------------------------------
Operation:          [QUERY]
--------------------------------------------------
Querying stress type: INVALIDATE_INTERNAL_CACHE_CQ              -
ENABLED
Querying stress type: INVALIDATE_INTERNAL_CACHE_GENERAL_1       -
ENABLED
Querying stress type: INVALIDATE_INTERNAL_CACHE_QP              -
ENABLED
Querying stress type: INVALIDATE_INTERNAL_CACHE_RX_1            -
ENABLED
Querying stress type: INVALIDATE_INTERNAL_CACHE_RX_2            -
ENABLED
Querying stress type: INVALIDATE_INTERNAL_CACHE_RX_3            -
ENABLED
Querying stress type: INVALIDATE_INTERNAL_CACHE_RX_4            -
ENABLED
Querying stress type: INVALIDATE_INTERNAL_CACHE_RX_5            -
NOT SUPPORTED
Querying stress type: INVALIDATE_INTERNAL_CACHE_SX1             -
ENABLED
Querying stress type: INVALIDATE_INTERNAL_CACHE_SX2             -
ENABLED
Querying stress type: INVALIDATE_INTERNAL_CACHE_SX3             -
ENABLED
```

Hang Types

ConnectX-4/ConnectX-4 Lx/ConnectX-5 Adapter Cards Hang Types

The following are the hang types available for ConnectX-4/ConnectX-4 Lx/ConnectX-5 adapter cards:

Category	Stress Type	Description	Notes
Hang FW/HW	FFSER	Initialize FaultInjector object	• This hang type is supported in BlueField-2 device only. • No extra flag is required.
	STOP_RX_PER_PRIO[1]		This type requires the following extra flags: • vl_mask - 16 bit • port_num - 8 bit

```
mlxfwstress -d  mt4115_pciconf0 -o on --hang-type
STOP_RX_PER_PRIO --extra %STOP_RX_PER_PRIO[0x00100FF]
Random seed: [1588056318]
-------------------------------------------------
Operation:         [ON]
-------------------------------------------------
Turning ON hang type: STOP_RX_PER_PRIO                          -
PASSED
```

To turn this Hang Type, the command must be executed in the following format:

Example:

```
mlxfwstress -d mt4115_pciconf0 -o on --hang-type STOP_RX_PER_PRIO
--extra % STOP_RX_PER_PRIO [0x000100FF]
output:
Random seed: [1573642282]
-------------------------------------------------
Operation: [ON]
-------------------------------------------------
Turning ON hang type: STOP_RX_PER_PRIO-PASSED
```

Turning On Hang Types

To turn on a specific hang type:

```
mlxfwstress -d mt4103_pciconf0 -o on --hang-type HANG_SX1
--------------------------------------------------
Operation: [ON]
--------------------------------------------------
Turning ON hang type: Sx1 -PASSED
```

To turn on a set of hang types:

```
mlxfwstress -d mt4103_pciconf0 -o on --hang-type
HANG_SX1,HANG_RX1
--------------------------------------------------
Operation: [ON]
--------------------------------------------------
Turning ON hang type: Sx1 -PASSED
Turning ON hang type: Rx1 -PASSED
```

Turning Off Hang Types

To turn off a specific hang type:

```
mlxfwstress -d mt4103_pciconf0 -o off --hang-type HANG_SX1
--------------------------------------------------
Operation: [OFF]
--------------------------------------------------
```

```
Turning OFF hang type: Sx1 -PASSED
```

To turn off a set of hang types:

```
mlxfwstress -d mt4103_pciconf0 -o off --hang-type
HANG_SX1,HANG_RX1
--------------------------------------------------
Operation: [OFF]
--------------------------------------------------
Turning OFF hang type: Sx1 -PASSED
Turning OFF hang type: Rx1 -PASSED
```

Querying the Hang Types

To query the state of all hang types:

```
mlxfwstress -d mt4103_pciconf0 -o query --hang-type ALL
--------------------------------------------------
Operation: [QUERY]
--------------------------------------------------
Querying hang type: Sx1 -ENABLED
Querying hang type: Rx1 -ENABLED
Querying hang type: Tx -ENABLED
Querying hang type: Rx -ENABLED
```

Clearing all Stress/Hang Types

To clear all stress/hang types:

```
mlxfwstress - d mt4103_pciconf0 -o clear_all
-------------------------------------------------
Operation: [CLEAR_ALL]
-------------------------------------------------
Turning OFF hang type: Sx1 -PASSED
Turning OFF hang type: Rx1 -PASSED
Turning OFF hang type: Tx -PASSED
Turning OFF hang type: Rx -PASSED
Turning OFF stress type: stop_ce_instage_eqe -PASSED
Turning OFF stress type: stop_sxp_vl_arb_port1 -PASSED
Turning OFF stress type: stop_sxp_vl_arb_port2 -PASSED
Turning OFF stress type: stop_edbh -PASSED
Turning OFF stress type: stop_idbh -PASSED
Turning OFF stress type: stop_qpc_miss_machine_0 -PASSED
Turning OFF stress type: stop_qpc_miss_machine_1 -PASSED
Turning OFF stress type: stop_qpc_miss_machine_2 -PASSED
Turning OFF stress type: stop_qpc_miss_machine_3 -PASSED
Turning OFF stress type: lock_cegw -PASSED
Turning OFF stress type: lock_obgw_tpt -PASSED
Turning OFF stress type: lock_obgw_tcu -PASSED
Turning OFF stress type: lock_obgw_sxd -PASSED
Turning OFF stress type: lock_qpcgw_rx -PASSED
Turning OFF stress type: lock_semaphore_ipc_sx1 -PASSED
Turning OFF stress type: lock_semaphore_ipc_rx0 -PASSED
Turning OFF stress type: lock_semaphore_ipc_rx1 -PASSED
Turning OFF stress type: lock_semaphore_ipc_ldb -PASSED
Turning OFF stress type: invalidate_caches -PASSED
```

Clearing the Semaphore

To clear the semaphore:

```
mlxfwstress -d mt4103_pciconf0 -o clear_semaphore
```

```
-------------------------------------------------
Operation: [CLEAR_SEMAPHORE]
-------------------------------------------------
Semaphore was cleared successfully
```

Random Operation

There are two random modes you can choose from:

- Single - gives a set of stress types, in each iteration one stress type is chosen an toggled ON/OFF according to his current state
- Wild - gives a set of stress types, in each iteration a random subset of stress types is chosen and toggled ON/OFF according to their current state

Setting the Random Mode for the Stress Types

To set the Single Mode:

```
mlxfwstress -d mt4103_pciconf0 -o random --rand-mode single -t
STOP_CE_INSTAGE_EQE --stress-delay 0.2 --iterations 10
-------------------------------------------------
Operation: [RANDOM]
-------------------------------------------------
#############################################
Random:
Iterations delay: 0.2 [sec]
Iterations number: 10
Max on time: 1 [sec]
#############################################
RANDOM ITERATION: [1]
[stop_ce_instage_eqe]: [ON] , duration since last operation: 0
[ms]
RANDOM ITERATION: [2]
[stop_ce_instage_eqe]: [OFF], duration since last operation: 200
[ms]
```

```
RANDOM ITERATION: [3]
[stop_ce_instage_eqe]: [ON] , duration since last operation: 201 [ms]
RANDOM ITERATION: [4]
[stop_ce_instage_eqe]: [OFF], duration since last operation: 200 [ms]
RANDOM ITERATION: [5]
[stop_ce_instage_eqe]: [ON] , duration since last operation: 200 [ms]
RANDOM ITERATION: [6]
[stop_ce_instage_eqe]: [OFF], duration since last operation: 201 [ms]
RANDOM ITERATION: [7]
[stop_ce_instage_eqe]: [ON] , duration since last operation: 200 [ms]
RANDOM ITERATION: [8]
[stop_ce_instage_eqe]: [OFF], duration since last operation: 201 [ms]
RANDOM ITERATION: [9]
[stop_ce_instage_eqe]: [ON] , duration since last operation: 200 [ms]
Turning OFF stress type: stop_ce_instage_eqe
RANDOM ITERATION: [10]
[stop_ce_instage_eqe]: [ON] , duration since last operation: 200 [ms]
=======================================================
Turning off all stress types after random:
Turning OFF stress type: stop_ce_instage_eqe
```

- As seen in the example above, after the specified number of iterations, the tool turns off all the stress types.
- The default value for stress-delay is 1 second.
- If no number of iterations was supplied then the user is expected to stop the tool with ctrl+c. The tool turns off all the stress types.

To set the Wild Mode:

```
mlxfwstress -d mt4103_pciconf0 -o random --rand-mode wild -t ALL --stress-delay 0.2 --max-rand-on 1 --iterations 5
-------------------------------------------------
Operation: [RANDOM]
-------------------------------------------------
#############################################
Random:
Iterations delay: 0.2 [sec]
Iterations number: 5
Max on time: 1 [sec]
#############################################

RANDOM ITERATION: [1]
[stop_ce_instage_eqe]: [ON] , duration since last operation: 0 [ms]
[stop_sxp_vl_arb_port2]: [ON] , duration since last operation: 0 [ms]
[stop_edbh]: [ON] , duration since last operation: 0 [ms]
[stop_idbh]: [ON] , duration since last operation: 0 [ms]
[stop_qpc_miss_machine_0]: [ON] , duration since last operation: 0 [ms]
[stop_qpc_miss_machine_3]: [ON] , duration since last operation: 0 [ms]
[lock_cegw]: [ON] , duration since last operation: 0 [ms]
[lock_obgw_tcu]: [ON] , duration since last operation: 0 [ms]
[lock_qpcgw_rx]: [ON] , duration since last operation: 0 [ms]
[lock_semaphore_ipc_sx1]: [ON] , duration since last operation: 0 [ms]

RANDOM ITERATION: [2]
[stop_sxp_vl_arb_port1]: [ON] , duration since last operation: 0 [ms]
[stop_edbh]: [OFF], duration since last operation: 203 [ms]
```

```
[stop_idbh]: [OFF], duration since last operation: 203 [ms]
[stop_qpc_miss_machine_3]: [OFF], duration since last operation:
202 [ms]
[lock_cegw]: [OFF], duration since last operation: 202 [ms]
[lock_obgw_tpt]: [ON] , duration since last operation: 0 [ms]
[lock_obgw_tcu]: [OFF], duration since last operation: 203 [ms]
[lock_semaphore_ipc_rx0]: [ON] , duration since last operation: 0
[ms]
[lock_semaphore_ipc_rx1]: [ON] , duration since last operation: 0
[ms]
[lock_semaphore_ipc_ldb]: [ON] , duration since last operation: 0
[ms]

RANDOM ITERATION: [3]
[stop_ce_instage_eqe]: [OFF], duration since last operation: 406
[ms]
[stop_sxp_vl_arb_port2]: [OFF], duration since last operation:
406 [ms]
[stop_edbh]: [ON] , duration since last operation: 203 [ms]
[stop_idbh]: [ON] , duration since last operation: 203 [ms]
[stop_qpc_miss_machine_0]: [OFF], duration since last operation:
406 [ms]
[stop_qpc_miss_machine_2]: [ON] , duration since last operation:
0 [ms]
[lock_obgw_tpt]: [OFF], duration since last operation: 203 [ms]
[lock_obgw_sxd]: [ON] , duration since last operation: 0 [ms]
[lock_semaphore_ipc_sx1]: [OFF], duration since last operation:
405 [ms]
[lock_semaphore_ipc_ldb]: [OFF], duration since last operation:
203 [ms]

RANDOM ITERATION: [4]
[stop_sxp_vl_arb_port2]: [ON] , duration since last operation:
203 [ms]
[stop_edbh]: [OFF], duration since last operation: 202 [ms]
[stop_idbh]: [OFF], duration since last operation: 202 [ms]
```

```
[stop_qpc_miss_machine_1]: [ON] , duration since last operation:
0 [ms]
[stop_qpc_miss_machine_3]: [ON] , duration since last operation:
406 [ms]
[lock_obgw_tpt]: [ON] , duration since last operation: 202 [ms]
[lock_obgw_tcu]: [ON] , duration since last operation: 406 [ms]
[lock_obgw_sxd]: [OFF], duration since last operation: 203 [ms]
[lock_semaphore_ipc_sx1]: [ON] , duration since last operation:
203 [ms]
[lock_semaphore_ipc_rx1]: [OFF], duration since last operation:
406 [ms]
[invalidate_caches]: [ON] , duration since last operation: 0 [ms]

Turning OFF stress type: stop_sxp_vl_arb_port1
Turning OFF stress type: stop_sxp_vl_arb_port2
Turning OFF stress type: stop_qpc_miss_machine_1
Turning OFF stress type: stop_qpc_miss_machine_2
Turning OFF stress type: stop_qpc_miss_machine_3
Turning OFF stress type: lock_obgw_tpt
Turning OFF stress type: lock_obgw_tcu
Turning OFF stress type: lock_qpcgw_rx
Turning OFF stress type: lock_semaphore_ipc_sx1
Turning OFF stress type: lock_semaphore_ipc_rx0
Turning OFF stress type: invalidate_caches

RANDOM ITERATION: [5]
[stop_sxp_vl_arb_port2]: [ON] , duration since last operation:
202 [ms]
[stop_idbh]: [ON] , duration since last operation: 322 [ms]
[lock_obgw_tpt]: [ON] , duration since last operation: 202 [ms]
[lock_obgw_tcu]: [ON] , duration since last operation: 202 [ms]
[lock_qpcgw_rx]: [ON] , duration since last operation: 202 [ms]
[invalidate_caches]: [ON] , duration since last operation: 202
[ms]
=======================================================
Turning off all stress types after random:
```

```
Turning OFF stress type: stop_sxp_vl_arb_port2
Turning OFF stress type: stop_idbh
Turning OFF stress type: lock_obgw_tpt
Turning OFF stress type: lock_obgw_tcu
Turning OFF stress type: lock_qpcgw_rx
Turning OFF stress type: invalidate_caches
```

ConnectX-3/ConnectX-3 Pro Adapter Cards Hang Types

The following are the hang types available for ConnectX-3/ConnectX-3 Pro adapter cards:

Category	Stress Type	Description	Notes
Hang FW/HW	HANG_SX1		
	HANG_RX1		
	HANG_TX		
	HANG_RX		
	ALL		Hang types that require extra flags are not supported when running with the 'ALL' option.

resourcedump Utility

The resourcedump tool extracts and prints data segments generated by the firmware. It is supported in 5th generation NIC's devices. The dump output is used by NVIDIA for debug and troubleshooting.

> **Note**
>
> Scapy and Pyverbs are no longer resourcedump dependencies. The same functionality has been changed with a C code that is based on

the RDMA core included in OFED (v5.7 and up) or Upstream (Kernel v5.14 and up).

Note

mstresourcedump can be used only if Python 3.x is installed. Using lower versions will result in tool's failure.

Note

To use the memory mode, the user must install the MLNX_OFED driver version that includes the rdma-core package, and in addition, must install the scapy package.

Note

It is important for the user to generate a bin file for debugging and troubleshooting cases when needed by NVIDIA.

Note

If the firmware version used is not supported, the tool will generate the following error message:

```
"Error: Failed to fetch query data with exception:
Failed to send Register RESOURCE DUMP with rc: 515.
Exiting...".
```

resourcedump Usage

```
resourcedump [-h] [-v] {dump,query}
```

where

dump	Dump command
query	Query command
-h, --help	Show help message and exit
-v, --version	Shows tool version and exit

resourcedump query Usage

```
resourcedump query [-h] [-v VIRTUAL_HCA_ID] [-m [MEM]] -d DEVICE
```

where

-h, --help	Show help message and exit
-v, --virtual-hca-id	The virtual HCA (host channel adapter, NIC) ID
-d, --device	The device name
-m, --mem	Perform the dump through memory (OFED with rdma-core dependency). Optionally accepts: [rdma device (for example "mlx5_4")]

An example of how to run the query command:

```
# resourcedump query --device /dev/mst/mt4119_pciconf0
-------------------------------------------------------------------
                 Segment Type - 0x1300 (FULL_EQC)
```

```
Dump Params                          Applicability    Special
Values
--------------------------------   --------------   -------------
-
index1 (EQN)                         Mandatory        N/A
num-of-obj1                          N/A              N/A
index2 (N/A)                         N/A              N/A
num-of-obj2                          N/A              N/A
-------------------------------------------------------------------
-------------------------------------------------------------------

                 Segment Type - 0x1000 (FULL_QPC)
Dump Params                          Applicability    Special
Values
--------------------------------   --------------   -------------
-
index1 (QPN)                         Mandatory        N/A
num-of-obj1                          N/A              N/A
index2 (N/A)                         N/A              N/A
num-of-obj2                          N/A              N/A
…
…
…
```

resourcedump dump Usage

```
usage: resourcedump dump [-h] -d DEVICE -s SEGMENT [-v
VIRTUAL_HCA_ID] [-i1 INDEX1]
[-i2 INDEX2][-n1 NUM_OF_OBJ1] [-n2 NUM_OF_OBJ2] [-de DEPTH] [-b
BIN] [-m [MEM]]
```

where

-h, --help	Show help message and exit
-v, --virtual-hca-id	The virtual HCA (host channel adapter, NIC) ID
-i1, --index1	The first context index to dump (if supported for this segment)
-i2, --index2	The second context index to dump (if supported for this segment)
-n1, --num-of-obj1	The number of objects to be dumped (if supported for this segment). accepts: ["all", "active", number, depends on the capabilities]
-n2, --num-of-obj2	The number of objects to be dumped (if supported for this segment). accepts: ["all", "active", number, depends on the capabilities]
-de, --depth	The depth of walking through reference segments. 0 stands for flat, 1 allows crawling of a single layer down the struct, etc. "inf" for all
-b, --bin	The output to a binary file that replaces the default print in hexadecimal, a readable format
-d, --device	The device name
-s, --segment	The segment to dump
-m, --mem	Perform the dump through memory (OFED with rdma-core dependency). Optionally accepts: [rdma device (for example "mlx5_4")]

Examples of how to:

- Run the dump command:

```
# resourcedump dump --device /dev/mst/mt4119_pciconf0 --
segment 0x1200 --index1 0x404 --depth 0
Found 10 segments:
--------------------------------------------
Segment Type: 0xfffe
Segment Size: 16 Bytes
Segment Data:
0x0004FFFE 0x00000000 0x00000000 0x101A0111
--------------------------------------------
Segment Type: 0xfffa
Segment Size: 20 Bytes
Segment Data:
```

```
0x0005FFFA 0x12000000 0x00000404 0x00000000
0x00000000
-----------------------------------------
```

- Run the Dump command and save it in bin file:

```
# resourcedump dump --device /dev/mst/mt4119_pciconf0 --
segment 0x1200 --index1 0x404 --depth 0 --bin segment_1200.bin
write to file:  segment_1200.bin
```

resourceparse Utility

The resourceparse tool parses and prints data segments content. The parser's output is used by NVIDIA representatives for debugging and troubleshooting.

This tool parses the output of the "resourcedump" tool. There are several parsing methods (see --parser):

- ABD:

  This parsing method receives an ADB file and parses each segment data according to the ADB layout of the node with the segment_id attribute corresponding to the segment.

- Map:

  This parsing method assumes that the provided resource-segments represent a memory map by DWORD pairs of address-value, and outputs each pair in a new line (similar to mstdump).

> **Note**
>
> The tool applicable inputs for parsing can be the resourcedump outputs (bin file or the “human readable” format”), or the devlink json

format output.

> **Note**
>
> To parse the segments data in the most efficient way, the user must use the most suitable adb file. For the adb file, please contact Support.

resourceparse Usage

```
resourceparse -d DUMP_FILE [-p RESOURCE_PARSER] [--version] [-o OUT] [-v] -a ADB_FILE [-r]
```

where

-d, --dump-file	Location of the dump file used for parsing
-p, --parser	RESOURCE_PARSER: Available options: ['adb', 'map']. Default: 'adb'. see Parsing Methods.
-a, --adb-file	Location of the ADB file
-h, --help	Shows this help message and exit
--version	Shows the tool's version and exit
-o, --out	Location of the output file
--out-dir	Location of the output directory, a separate file is created for each segment
-r, --raw	Prints the raw data in addition to the parsed data
--hide-segment-header	Hide segment header during printing
-v	Verbosity notice

Examples:

- How to run basic parsing:

```
# resourceparse --dump-file notice.txt --adb-file fw-4119-
rel-16_23_2008.adb

Parse 4 segments:

--------------------------------------------------------------
-------------------

                    Segment - segment_info (0xfffe)

segment_header.segment_type = 0xfffe

segment_header.length_dw = 0x4

dump_version = 0x0

hw_version = 0x0

fw_version = 0x1063232c

--------------------------------------------------------------
-------------------

                    Segment - segment_command (0xfffa)

segment_header.segment_type = 0xfffa

segment_header.length_dw = 0x5

vhca_id = 0x0

segment_called = 0x2000
```

```
index1 = 0x21

index2 = 0x0

num_of_obj1 = 0x0

num_of_obj2 = 0x0

-------------------------------------------------------------
-------------------

                    Segment - segment_notice (0xfff9)

segment_header.segment_type = 0xfff9

segment_header.length_dw = 0xc

syndrome_id = 0x211

notice[0] = 0x2000

notice[1] = 0x21

notice[2] = 0x0

notice[3] = 0x0

notice[4] = 0x496e7661

notice[5] = 0x6c696420

notice[6] = 0x52657300

notice[7] = 0x0

notice msg =  !Invalid Res
```

```
--------------------------------------------------------------------------------
                    Segment - segment_terminate (0xfffb)
segment_header.segment_type = 0xfffb
segment_header.length_dw = 0x1
--------------------------------------------------------------------------------
```

- How to run parsing with 'raw' and 'verbosity' options:

```
# resourceparse --dump-file notice.txt --adb-file fw-4119-rel-16_23_2008.adb –raw -v
Notice - adb fw version 16.23.2008 is used for parsing while dump fw version is 16.99.9004
Parse 4 segments:
--------------------------------------------------------------------------------
                    Segment - segment_info (0xfffe)
segment_header.segment_type = 0xfffe
segment_header.length_dw = 0x4
dump_version = 0x0
```

```
hw_version = 0x0

fw_version = 0x1063232c

RAW DATA:

DWORD [0-3]     :0x0004FFFE  0x00000000  0x00000000  0x1063232C

-------------------------------------------------------------
-------------------

                    Segment - segment_command (0xfffa)

segment_header.segment_type = 0xfffa

segment_header.length_dw = 0x5

vhca_id = 0x0

segment_called = 0x2000

index1 = 0x21

index2 = 0x0

num_of_obj1 = 0x0

num_of_obj2 = 0x0

RAW DATA:

DWORD [0-3]     :0x0005FFFA  0x20000000  0x00000021  0x00000000

DWORD [4]       :0x00000000
```

```
--------------------------------------------------------------
-------------------
                    Segment - segment_notice (0xfff9)
segment_header.segment_type = 0xfff9
segment_header.length_dw = 0xc
syndrome_id = 0x211
notice[0] = 0x2000
notice[1] = 0x21
notice[2] = 0x0
notice[3] = 0x0
notice[4] = 0x496e7661
notice[5] = 0x6c696420
notice[6] = 0x52657300
notice[7] = 0x0
RAW DATA:
DWORD [0-3]     :0x000CFFF9 0x00000211 0x00000000 0x00000000
DWORD [4-7]     :0x00002000 0x00000021 0x00000000 0x00000000
DWORD [8-11]    :0x496E7661 0x6C696420 0x52657300 0x00000000
notice msg =  !Invalid Res
```

```
--------------------------------------------------------------
-------------------

                    Segment - segment_terminate (0xfffb)

segment_header.segment_type = 0xfffb

segment_header.length_dw = 0x1

RAW DATA:

DWORD [0]       :0x0001FFFB

--------------------------------------------------------------
-------------------
```

- How to run parsing and save it into a file:

```
# resourceparse --dump-file notice.txt --adb-file fw-4119-
rel-16_23_2008.adb –out out_flie.txt

write to file:  out_flie.txt
```

stedump Utility

The stedump tool is a packet simulator for host NIC steering solutions. It is supported in 5th generation NIC devices and only when using Python v3.4 and above. The dump output of HW steering is used for debugging and troubleshooting.

Prerequisites

Using the MFT with the --with-pcap option to install stedump utility requires the following third-party dependencies:

- Libraries and header files for the libpcap library
- Libraries and header files for Python development library
- Package Installer for Python (PIP) available

stedump Usage

```
stedump [-h] --read-file <read_file> [--packet-format
<packet_format>] [--output-file <output_file>] [--verbosity
<verbosity>] [-v] {live,offline}
```

Where

-h, --help	Show help message and exit
--version	Show version information and exit
--packet-format	Specifies the I/O packet file format (default: hex)
--output-file	Redirect the output to specific file (default: stdout)
--verbosity	Increase output verbosity (default: 0)
--read-file	Specifies the packet(s) filename to read from
live	Run in live mode.
offline	Run in offline mode.

There are three kind of packet formats:

1. hex - Represents one or more packets separated by a newline in hexadecimal format.
2. pcap - Represents one or more packets in packet capture format.
3. raw - Represents a single packet in binary data format.

Note

The pcap packet format is not supported by default, and requires installing MFT with the --with-pcap option.

Note

Offline mode is not supported.

stedump live Usage

```
stedump live [-h] --device <device> --port <physical_port>
{egress,ingress}
```

Where

-h, --help	Show help message and exit
-d , --device	Perform operation for a specified MST device
--port	Specifies the physical port number
egress	Specifies the packet source to be egress for TX flows
ingress	Specifies the packet source to be ingress for RX flows

stedump live egress Usage

```
stedump live egress [-h] [--reg-a <reg_a_value>] [--virtual-hca-
id <virtual_hca_id>] [--sqn <sqn>] [--force-loopback] [--special-
root]
```

Where

-h, --help	Show help message and exit
--reg-a	Specifies steering register A value (default: 0)
--virtual-hca-id	Specifies the source virtual HCA ID (default: 0)
--sqn	Specifies the send queue context number (default: 0)
--force-loopback	specifies whether to use QP force loopback
--special-root	specifies whether to use QP special root

An example of how to run the egress (TX) packet flow:

```
# stedump --read-file tcp.hex live --device
/dev/mst/mt4119_pciconf0 --port 0 egress

PACKET_DATA

C0 C1 C2 C3 C4 C5 A0 A1 A2 A3 A4 A5 08 00 45 00

00 28 00 00 40 00 40 06 34 CB 01 01 01 01 02 02

02 02 10 E1 22 3D 00 00 BA BA 00 00 DE DA 51 23

FF FF AD 29 00 00



STE_MASKED[1] OUTER: source_qp: 0x40

[HIT] - hit_ix=0x6e polarity



STE_MASKED[2] OUTER: encapsulation_type: ROCE
```

```
[HIT] - hit_ix=0x6d polarity


STE_MASKED[3] : Always Hit

[HIT] - hit_ix=0x2000001a


STE_MASKED[4] : Always Hit

[HIT] - hit_ix=0xf0000000


STE_MASKED[5] : Always Hit

[HIT] - hit_ix=0xf0000009


STE_MASKED[6] OUTER: dmac: c0:c1:c2:c3:c4:c5, l3_type: IPV4

[HIT] - hit_ix=0xf000000e


STE_MASKED[7] OUTER: sip: 1.1.1.1

[ACTION] - COUNT { flow_counter_id=0x801199, gvmi=0x0 }

[ACTION] - MODIFY_HEADER { number_of_re_write_actions:11, ix=0x57
}

[HIT] - hit_ix=0x1a44c
```

```
STE_MASKED[8] : Always Hit

[ACTION] - COUNT { flow_counter_id=0x2a, gvmi=0x0 }

[ACTION] - WIRE {  }

[HIT] - hit_ix=0x0

[ACTION] - SX TERMINATOR { WIRE  }

- STEERING_HOPS - 8
```

An example of how to run the ingress (RX) packet flow:

```
# stedump --read-file tcp.hex live --device
/dev/mst/mt4119_pciconf0 --port 0 ingress

PACKET_DATA

E0 E1 E2 E3 E4 E5 A0 A1 A2 A3 A4 A5 08 00 45 00

00 5E 00 00 40 00 40 11 34 8A 01 01 01 01 02 02

02 02 04 D2 17 C1 00 4A DC C1 01 80 65 58 CC CE

23 00 61 61 61 61 B0 B1 B2 B3 B4 B5 A0 A1 A2 A3

A4 A5 08 00 45 00 00 28 00 00 40 00 40 06 34 CB

01 01 01 01 02 02 02 02 10 E1 22 3D 00 00 BA BA
```

```
00 00 DE DA 51 23 FF FF AD 29 00 00



STE_MASKED[1] OUTER: encapsulation_type: ROCE

[ACTION] - COUNT { flow_counter_id=0x29, gvmi=0x0 }

[HIT] - hit_ix=0x2000001a polarity



STE_MASKED[2] : Always Hit

[HIT] - hit_ix=0xf0000000



STE_MASKED[3] : Always Hit

[HIT] - hit_ix=0xf0000009



STE_MASKED[4] OUTER: dmac: e0:e1:e2:e3:e4:e5, l3_type: IPV4

[HIT] - hit_ix=0xf000000f



STE_MASKED[5] OUTER: sip: 1.1.1.1

[ACTION] - COUNT { flow_counter_id=0x801199, gvmi=0x0 }

[ACTION] - DECAP { L2 }
```

```
[ACTION] - MODIFY_HEADER { number_of_re_write_actions:1, ix=0xd7 }

[HIT] - hit_ix=0xa00001a5



STE_MASKED[6] : Always Hit

[HIT] - hit_ix=0xa00001a2



STE_MASKED[7] : Always Hit

[ACTION] - QP { gvmi=0x0,qp=0x108d }

[HIT] - hit_ix=0x0

[ACTION] - RX TERMINATOR {  }

- STEERING_HOPS - 7
```

Cable Utilities

MFT can work against the cables that are connected to the devices on the machine, or in the InfiniBand fabric in the following scenarios:

- Accessing the cable using the local PCI device. Supported in:
    - ConnectX-4 and newer devices
    - All the platforms
- Accessing the cable using the IB fabric. Supported in:

- All the devices
- Linux and Windows since IB driver is required
- Supported cables are all QSFP and SFP

Cable Discovery

How to Discover the Cables

To discover the cables that are connected to the local devices:

```
mst cable add
```

This command will scan all the local PCI devices and try to discover cable connected to each port.

To expand the discovery to include also the IB fabric, use the "--with_ib" flag. This flag by default will scan all the ib devices from the ibstat/ibv_devices output. To run only a specific interface and port, the interface or the port should be specified after the flag.

Examples:

To scan all the fabric:

```
mst cable add --with_ib
```

To scan a specific interface:

```
mst cable add --with_ib mlx4_0
```

or:

```
mst cable add --with_ib mlx4_0 1
```

Representing the Cables in mst Status

Local Cables

The name of the cable will be the same name as the mst-device/PCI-device with _cable_<port>.

Examples:

```
mst cable add
-I- Added 2 cable devices.


mst status
MST modules:
…
Cables:
-------------------
mt4115_pciconf0_cable_0
mt4115_pciconf0.1_cable_1
```

Remote Cables

When using the '--with_ib' flag, the name of the cable devices are created the same as the Inband devices with _cable.

```
> mst cable add --with_ib
-I- Added 4 cable devices ..
> mst status
```

```
Cables:
--------------------
CA_MT4113_HCA-4_lid-0x0002,mlx5_0,1_cable
CA_MT4115_HCA-2_lid-0x0001,mlx5_0,1_cable
mt4115_pciconf0_cable_0
mt4115_pciconf0_cable_1
```

Working with Cables

The following are the tools that can work with cables: mstdump, mlxdump and mlxcables.

The below are examples using the tools mentioned above.

mstdump

```
mstdump mt4115_pciconf0_cable_0
0x00000000 0x0002060d
0x00000004 0x00000000
0x00000008 0x00000000
0x0000000c 0x00000000
0x00000010 0x00000000
0x00000014 0x4a340000
0x00000018 0x8b7f0000
0x0000001c 0x00000000
0x00000020 0xc0210000
0x00000024 0x901aa61d
0x00000028 0x6dc9521c
0x0000002c 0xe2d12fca
…
…
…
```

mlxdump

```
# mlxdump -d mt4115_pciconf0_cable_0 snapshot
-I- Dumping crspace...
-I- crspace was dumped successfully
-I- Dump file "mlxdump.udmp" was generated successfully
```

For the mlxcables example, refer to section mlxcables.

mlxcables - Cables Tool

Note

- mlxcables is used for R/W to cable EEPROM
- The information of the devices will be collected by PRM command from switch Asic by mlxlink tool only
- mlxcables will continue to support R/W operations to EEPROM, and will not be deprecated currently due to Independent module. Meaning, new features will not be supported in mlxcables

The mlxcables tool allows users to access the cables and do the following:

- Query the cable and get its IDs
- Read specific addresses in the EEPROM
  - Read a specific register by its name. Supported registers are received by the tool (depends on the cable type)
  - Dump all the cable EEPROM bytes in RAW format
  - Upgrade the FW image on the cable uC (Only on cables that support ISSU)

mlxcables Synopsis

```
[-d|--dev <DeviceName>] [-h|--help] [-v|--version] [-q|--query]
[--DDM] [-r|--read] [--print_raw] [--dump] [-b|--bytes_line
<bytesPerLine>] [-p|--page <pageNum>] [-o|--offset <pageOffset>]
[-l|--length <length>] [-a|--address <address>] [-y|--yes] [--no]
[--read_reg <Register>] [--read_all_regs] [--show_all_regs] [--
customization <Customization_type>]
```

where:

-d|--dev <DeviceName>	Perform operation for specified cable
-h|--help	Show this message and exit
-v|--version	Show the executable version and exit
-q|--query	Query cable info
--DDM	Get cable DDM query
-r|--read	Read from cable
--print_raw	Print bytes in raw format
--dump	Dump all cable pages in RAW format
-b|--bytes_line <bytesPerLine>	Bytes per line in the raw print (multiples of 4, default: 4)
-p|--page <pageNum>	Specific Page number to do the read/write operation
-o|--offset <pageOffset>	Specific Page offset
-l|--length <length>	Length of the needed data in bytes to read (default: 1 Byte)
-a|--address <address>	Address (Replacement for page+offset)
--read_reg <Register>	Read register from cable
--read_all_regs	Read all registers from cable
--show_all_regs	Show all registers in the cable

--customization <customization_type>	Show cable specific customization

Notes:

- For QSFP transceivers, the tool reads the address from I2C address of 0x50. For further information, please see spec SFF8636.
- For SFP transceivers, the tool reads from I2C address 0x50 and names it page 0. When reading from I2C address 0x51 the pages will be read as page <x+1>, for example:
    - I2C address 0x51 page 0 will be referred in the tool as page 1.
    - I2C address 0x51 page 1 will be referred in the tool as page 2. For further information, please see spec SFF8472.

Examples:

To read specific byte/s in the cable pages:

```
mlxcables -d mt4115_pciconf0_cable_0 -r -p 0 -o 165 -l 3
Page[0].Byte[165] = 0x00
Page[0].Byte[166] = 0x02
Page[0].Byte[167] = 0xc9
```

Another way to read from a specific page is to use the '--address <ADDR>' flag where ADDR=0x<PAGE><OFFSET>, for example to read the same bytes with -a:

```
mlxcables -d mt4115_pciconf0_cable_0 -r -a 0x00A5 -l 3
Page[0].Byte[165] = 0x00
Page[0].Byte[166] = 0x02
Page[0].Byte[167] = 0xc9
```

To read in raw format:

```
# mlxcables -d mt4115_pciconf0_cable_0 -r -p 0 -o 128 -l 12 --
print_raw

128: 0d 8c 23 81
132: 00 00 00 00
136: 00 00 00 05
```

To control Bytes per line, use -b:

```
# mlxcables -d mt4115_pciconf0_cable_0 -r -p 0 -o 128 -l 12 --
print_raw -b 8
128: 0d 8c 23 81 00 00 00 00
136: 00 00 00 05
```

To query the cable:

```
mlxcables -d mt4115_pciconf0_cable_0 -q
Cable name     : mt4115_pciconf0_cable_0
FW version     : 2.2.550
FW Dev ID      : 0x21
FW GW version : Legacy
-------- Cable EEPROM --------
Identifier     : QSFP+ (0dh)
Technology     : 1550 nm DFB (50h)
Compliance     : 40G Active Cable (XLPPI), 100G AOC or 25GAUI C2M
AOC. Providing a worst BER of 10^(-12) or below
Wavelength     : 1550 nm
OUI            : 0x0002c9
Vendor         : Mellanox
Serial number : MT1602FT00022
Part number    : MFS1200-E003
```

```
Revision        : AB
Temperature     : 54 C
Length          : 3 m
```

Get the DDM query of the cable:

```
# mlxcables -d mt4115_pciconf0_cable_0 --DDM
Cable DDM:
----------
Temperature : 37C
Voltage     : 3.3010V
RX Power    : -0.6712dBm
TX Power    : 0.8877dBm
TX Bias     : 6.7500mA
----- Flags -----
Temperature:
  Alarm high    : 0
  Warning high  : 0
  Warning low   : 0
  Alarm low     : 0
Voltage:
  Alarm high    : 0
  Warning high  : 0
  Warning low   : 0
  Alarm low     : 0
RX/TX Power and TX Bias:
  RX Power alarm high   : 0
  RX Power warning high: 0
  RX Power warning low  : 0
  RX Power alarm low    : 0
  TX Power alarm high   : 0
  TX Power warning high: 0
  TX Power warning low  : 0
  TX Power alarm low    : 0
```

```
    TX Bias alarm high    : 0
    TX Bias warning high : 0
    TX Bias warning low  : 0
    TX Bias alarm low    : 0
  ----- Thresholds -----
    Temperature high alarm threshold   : 80C
    Temperature high warning threshold : 70C
    Temperature low warning threshold  : 0C
    Temperature low alarm threshold    : -10C

    Voltage high alarm threshold   : 3.5000V
    Voltage high warning threshold: 3.4650V
    Voltage low warning threshold : 3.1350V
    Voltage low alarm threshold    : 3.1000V

    RX Power high alarm threshold : 5.3999dBm
    RX Power high warn threshold  : 2.4000dBm
    RX Power low warn threshold   : -10.3012dBm
    RX Power low alarm threshold  : -13.3068dBm

    TX Power high alarm threshold : 5.3999dBm
    TX Power high warn threshold  : 2.4000dBm
    TX Power low warn threshold   : -8.4013dBm
    TX Power low alarm threshold  : -11.4026dBm

    TX Bias high alarm threshold  : 8.5000mA
    TX Bias high warn threshold   : 8.0000mA
    TX Bias low warn threshold    : 6.0000mA
    TX Bias low alarm threshold   : 5.4920mA
```

To read by register name:

Get the list of the supported registers.

```
mlxcables -d mt4115_pciconf0_cable_0 --show_all_regs
Available registers per page:
====================================================
page00_high registers:
====================================================
identifier     |
ext_identifier |
connector_type |
..
..
..
vendor_oui     |
..
```

Read the register with the register name you choose (e.g. vendor_oui, identifier).

```
mlxcables -d mt4115_pciconf0_cable_0 --read_reg vendor_oui
vendor_oui = 0x0002c9
mlxcables -d mt4115_pciconf0_cable_0 --read_reg identifier
identifier = 0x0d
```

To read all the Eeprom of the cable:

```
mlxcables -d mt4115_pciconf0_cable_0 --read_all_regs
Available registers per page:
====================================================
page00_high registers:
====================================================
identifier | 0x0d
ext_identifier | 0x06
connector_type | 0x02
```

```
..
..
..
vendor_oui | 0x0002c9
..
```

This will print the same tables as "--show_all_regs" but with the data that was read.

To dump all the cable's pages in raw format:

```
# mlxcables -d mt4115_pciconf0_cable_0 --dump -b 16
+-----------------------------------------+
| Page: 0x00 , Offset: 000, Length: 0x80 |
+-----------------------------------------+
000: 0d 06 00 f0 00 00 00 00 00 00 00 00 00 00 00 00
016: 00 00 00 00 00 00 29 a7 00 00 80 7d 00 00 00 00
032: 00 00 37 fa 26 2a 33 68 1c 0c b3 f9 b2 76 c2 fc
048: c1 3e 00 00 00 00 00 00 00 00 00 00 00 00 00 00
064: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
080: 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00
096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 00
112: 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00
+-----------------------------------------+
| Page: 0x00 , Offset: 128, Length: 0x80 |
+-----------------------------------------+
128: 0d cc 23 81 00 00 00 00 00 00 00 05 ff 00 00 00
144: 00 00 03 50 4d 65 6c 6c 61 6e 6f 78 20 20 20 20
160: 20 20 20 20 1f 00 02 c9 4d 46 53 31 32 30 30 2d
176: 45 30 30 33 20 20 20 20 41 43 79 18 27 10 46 be
192: 18 07 f5 96 4d 54 31 36 31 33 46 54 30 30 36 39
208: 36 20 20 20 31 36 30 32 32 32 00 00 00 00 67 a9
224: 36 30 33 46 4d 41 32 30 33 47 31 34 32 38 20 20
240: 00 00 00 00 00 00 00 00 00 00 01 00 10 00 00 00
+-----------------------------------------+
```

```
| Page: 0x03 , Offset: 128, Length: 0x80 |
+----------------------------------------+
128: 50 00 f6 00 46 00 00 00 00 00 00 00 00 00 00 00
144: 88 b8 79 18 87 5a 7a 76 00 00 00 00 00 00 00 00
160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
176: 87 71 01 d3 43 e2 03 a5 00 00 00 00 00 00 00 00
192: 87 71 02 d4 43 e2 05 a5 00 00 00 00 00 00 00 00
208: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
224: a7 03 00 00 00 00 00 00 00 00 00 00 00 77 77 11 11
240: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

MTUSB Cable Board

The mlxcables tool supports reading cable data directly via i2c when the cable is connected to a dedicated board. The board is connected to the host with an MTUSB adapter.

Examples on a Windows machine:

After adding the cables using 'mst cable add' the following mst status is presented:

```
mst status
MST devices:
------------
  mtusb-1
Cable MST devices:
-------------------
  mtusb-1_cable
```

Query the cable:

```
mlxcables -d mtusb-1_cable
Querying Cables ....
```

```
Cable #1:
---------
Cable name     : mtusb-1_cable
FW version     : 2.0.208
FW Dev ID      : 0x20
FW GW version : Legacy
-------- Cable EEPROM --------
Identifier     : QSFP+ (0dh)
Technology     : 850 nm VCSEL (00h)
Compliance     : 40G Active Cable (XLPPI), 100G AOC (Active
Optical Cable) or 25GAUI C2M AOC.
Wavelength     : 850 nm
OUI            : 0x0002c9
Vendor         : Mellanox
Serial number : MT1707FT01544
Part number    : MFA1A00-E001
Revision       : A1
Temperature    : 31 C
Length         : 1 m
```

Read from a specific address:

```
mlxcables -d mtusb-1_cable -r -p 0 -o 165 -l 3
Page[0].Byte[165] = 0x00
Page[0].Byte[166] = 0x02
Page[0].Byte[167] = 0xc9
```

Cable Firmware Upgrade with Live-Patch

Live-Patch enables a seamless update of the cable firmware.

Prerequisites

The cable firmware gateway version must be set to ISFU (see "FW GW version : ISFU" in the example below).

Burn the cable firmware image binary.

```
# mlxcables -d mt52000_pciconf0_cable_22 -i img.bin -u
```

To get the cable's firmware binary, please contact Mellanox Support (support@mellanox.com).

Burning Process Example:

Query the device:

```
# mlxcables -d mt52000_pciconf0_cable_22
Querying Cables ....
Cable #1:
---------
Cable name     : mt52000_pciconf0_cable_22
FW version     : 32.20.121
FW Dev ID      : 0x22
FW GW version : ISFU
-------- Cable EEPROM --------
Identifier     : QSFP28 (11h)
Technology     : 850 nm VCSEL (00h)
Compliance     : Extended Specification Compliance is valid,
100GBASE-SR4 or 25GBASE-SR
Wavelength     : 850 nm
OUI            : 0x0002c9
Vendor         : Mellanox
Serial number : MT1623FT01610
Part number    : MMA1B00-C100D
Revision       : A2
```

```
Temperature    : 31 C
Length         : 50 m
```

Query the image:

```
# mlxcables -i img.bin
Image name     : img.bin
FW version     : 32.30.204
FW Dev ID      : 0x22
FW GW version : ISFU
```

Upgrade the firmware:

```
# mlxcables -i img.bin -d mt52000_pciconf0_cable_22 -u
Cable FW version : 32.20.121
Image FW version : 32.30.204
The image version is newer than the FW version on the cable, Do
you want to proceed (y/N)? y
100%
-I- The FW was updated successfully
```

Query and verify the new firmware version:

```
# mlxcables -d mt52000_pciconf0_cable_22
Querying Cables ....
Cable #1:
---------
Cable name     : mt52000_pciconf0_cable_22
FW version     : 32.30.204
FW Dev ID      : 0x22
FW GW version : ISFU
```

```
-------- Cable EEPROM --------
Identifier     : QSFP28 (11h)
Technology     : 850 nm VCSEL (00h)
Compliance     : Extended Specification Compliance is valid,
100GBASE-SR4 or 25GBASE-SR
Wavelength     : 850 nm
OUI            : 0x0002c9
Vendor         : Mellanox
Serial number  : MT1623FT01610
Part number    : MMA1B00-C100D
Revision       : A2
Temperature    : 31 C
Length         : 50 m
```

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 Support.

General Related Issues

Issue	Cause	Solution
Adapter is no longer identified by the operating system after firmware upgrade	Happens due to burning the wrong firmware on the adapter, firmware corruption or adapter's hardware failure.	Power cycle the server. If the issue persists, extract the adapter and contact Support
Server is booting in loop/not completing boot after performing adapter firmware upgrade	Happens due to burning the wrong firmware on the adapter, firmware corruption or adapter's hardware failure.	Extract the adapter and contact Support
Some of the 5th generation (Group II) devices are represented with only one mst device	For 5th generation (Group II) devices, there is only one method available for accessing the hardware. For example, Connect-IB device is	When querying a 5th generation (Group II) device, use the conf mst device (for example: dev/mst/mt4113_pciconfx)

Issue	Cause	Solution
(dev/mst/mt4113_pciconfx) in the output of mst status	represented bydev/mst/mt4113_pciconfx mst device	
Enabling hardware access after configuring new secure host key, fails	The new configuration of the secure host key was not loaded by the driver	Restart the driver before enabling the hardware access again
MFT tools fail on PCI device with the following errors: • Operation not permitted • Failed to identify device • Failed to detect device ID • Unknown device • No such device • Failed to open device	Tools PCI semaphore might be locked due to unexpected process shutdown.	Run the following command: # mcra -c <mst_pci_device> *Supported on MFT-4.4.0 and newer versions.

mlxconfig Related Issues

Issue	Cause	Solution
Server not booting after enabling SRIOV with high number of VFs	Setting number of VFs larger than what the Hardware and Software can support may cause the system to cease working	To solve this issue: 1. Disable SRIOV in bios 2. Reboot server 3. Change num of VFs 4. Enable SRIOV in bios
After resetting configuration using the tool on 5th generation (Group II) devices, the configuration's value does not change	Firmware loads the default configuration only upon reboot	Reboot the server

Installation Related Issues

Issue	Cause	Solution
Unable to install the tool package on ESXi platform and the following message is printed on the screen: Got no data from process	Insufficient privileges	1. Copy the tool's package to /tmp/vmware and continue with the installation. If the issue persists, reboot the ESX server and try again 2. Use full file path of the tool's package **Note**: an additional reboot will be required after completing the installation
Unable to install kernel-mft in Linux due to compilation error that contains the following message: 'error: conflicting types for 'compat_sigset_t''	CONFIG_COMPAT might not be enabled in the kernel configuration.	Set the CONFIG_COMPAT to “y” in the kernel .config file, and rebuild the kernel.

Firmware Burning Related Issues

Issue	Cause	Solution
The following message is printed on screen when performing firmware update: An update is needed for the flash layout. The operation is not failsafe and terminating the process is not allowed.	A flash alignment operation is required.	Approve the alignment, avoid process interrupt.
Firmware update fails with the following message: -E- Burning FS4 image failed: Bad parameter **Note:** This is a rare scenario.	Firmware compatibility issue.	Re-run the burn command with --no_fw_ctrl flag.
The following message is printed on screen when performing firmware update: Shifting between different image partition sizes requires current	Firmware compatibility issue.	Re-load firmware and re-run the burn command.

Issue	Cause	Solution
image to be re-programmed on the flash. Once the operation is done, reload FW and run the command again **Note:** This is a rare scenario.		
The following message is printed on screen when trying to query/burn a Connect-IB device: -E- Cannot open Device: /dev/mst/mt4113_pciconf0. B14 Operation not permitted MFE_CMDIF_GO_BIT_BUSY	Using an outdated firmware version with the Connect-IB adapter.	1. Unload MLNX_OFED driver: /etc/init.d/openibd stop. 2. Add “-ocr” option to the 'flint' command. For example: flint -d /dev/mst/mt4113_pciconf0 -ocr q
The following message is reported on screen when trying to remove the expansion ROM using the 'drom' option: -E- Remove ROM failed: The device FW contains common FW/ROM Product Version - The ROM cannot be removed separately.B9	Updating only the EXP_ROM (FlexBoot) for recent firmware images which requires adding the 'allow_rom_change' option.	Allow “-allow_rom_change” option to the “flint” command. For example: flint -d <mst_device> -allow_rom_change drom
Burning command fails and the following message is printed on screen: -E- Can not open 06:00.0: Can not obtain Flash semaphore (63). You can run "flint -clear_semaphore - d <device>" to force semaphore unlock. See help for details.	Semaphore can be locked for any of the following reasons: • Another process is burning the firmware at the same time • Failure in the firmware boot • Burning process was force- fully killed • In a Multi-Host environment, another Host is	If no other process is taking place at the same time run the following command: flint -d <device> -- clear_semaphore OR Reboot the machine.

Issue	Cause	Solution
	cur- rently burning the firmware	
Burning tool fails with the following message: –E– Unsupported binary version (2.0) please update to latest MFT package.	The binary version is incompatible with the burning tool.	Update MFT to the latest package.
mlxburn tool fails to generate a firmware image and displays the following message: –E– Unsupported MLX file version (2.0) please update to latest MFT package.	The MLX file version is incompatible with the image generation tool (mlxburn).	Update MFT to the latest package.
mlxburn tool fails to generate a firmware image and displays the following message -E- Perl Error: Image generation tool uses mic (tool) version 1.5.0 that is not supported for creating a bin file for this FW version. FW requires mic version 2.0.0 or above. Please update MFT package.	The MLX file version is incompatible with the image generation tool (mlxburn).	Update MFT to the latest package.
Burning tool fails with an error mentioning Firmware time stamping e.g -E- Burning FS3 image failed: Stamped FW version missmatch: 12.16.0212 differs from 12.16.0230	The device was set with a timestamp for a different firmware version than the one being burnt or the image is stamped with an older timestamp	Either set a newer timestamp on the image than there is on the device, or reset the timestamp completely. flint -d <device> ts reset flint -i <image> ts reset
Burning the image on Controlled FW (default update method: fw_ctrl in 'flint -d <device> query full' output), fails with: -E- Burning FS3 image failed: The Digest in the signature is wrong.	The image was changed without calculating the new digest on it with 'flint -i <img.bin> sign'.	Run 'flint -i <img.bin> sign', and retry.

Secure Firmware Related Issues

Issue	Cause	Solution
Changing device setting such as ROM/ GUIDS using the relevant flint commands result in failure with the following error: -E- <Operation> failed: Unsupported operation under Secure FW	Secure Firmware does not allow changes to the device data unless burning new Secure Firmware image.	N/A
Burning tool fails with the following error: -E- Burning FS3 image failed: The component is not signed.	The image is not signed with an RSA authentication.	Contact Support to receive a signed firmware image.
Burning tool fails with the following error: -E- Burning FS3 image failed: Rejected authentication.	The image authentication is rejected.	Contact Support to receive a signed firmware image.
Burning tool fails with the following error: -E- Burning FS3 image failed: Component is not applicable.	The image does not match the device (Wrong ID).	Contact Support to receive the firmware image for the device.
Burning tool fails with the following error: -E- Burning FS3 image failed: The FW image is not secured.	The image is not secured and is not accepted by the device.	Contact Support to receive a signed firmware image.
Burning tool fails with the following error: -E- Burning FS3 image failed: There is no Debug Token installed.	The debug firmware was burnt before the debug token was installed on the device.	Install the debug token using mlxconfig and then re-burn the firmware.
Burning firmware on a secure device fails with one of the following messages: • -E- Burning FS3 image failed: Rejected authentication • The FW image is not secured • The key is not applicable	The image was not secured in a the proper way.	Ask for a secure image with the right keys that match the device.
Secure Firmware fails when using flint brom and drom commands.	flint brom and drom commands are not	N/A

Issue	Cause	Solution
	supported.	
mlxdump and wqdump debug utilities do not work in Secure Firmware	A customer support token was not applied.	N/A
When the CR space is in read only mode, the tracers may demonstrate an unexpected behavior.	A writing permission is required for them to work properly.	N/A
Applying token on the device fails with one of the following messages: • Component is not applicable • The manufacturing base MAC was not listed • Mismatch FW version • Mismatch user timestamp • Rejected forbidden version	The token was not generated or signed in the proper way.	N/A
Burning the firmware using the "--use_dev_rom" flag has no effect and the ROM is replaced with the one on the image.	Controlled firmware does not support changing boot image component.	Use "--no_fw_ctrl".

Appendixes

- Assigning PSID
- Remote Access to NVIDIA Devices
- Booting HCA Device in Livefish Mode
- Burning a New Device
- Generating Firmware Secure and NV LifeCycle Configurations Files
- Updating Firmware Using ethtool/devlink and .mfa2 File
- Using MFT tools on Secured GPU Devices

Assigning PSID

In some cases, OEMs or board manufacturers may wish to use a specific FW configuration not supplied by NVIDIA. After setting the new FW parameters in an INI file, the user should assign a unique PSID (Parameter Set ID) to this new configuration. The PSID is kept as part of the FW image on the device NVMEM. The firmware burning tools use this field to retain FW settings while updating FW versions.

This appendix explains how to assign a new PSID for a user customized FW, and how to indicate to the burning tools that a new PSID exists.

> **Note**
>
> Please change FW parameters with caution. A faulty setting of FW parameters may result in undefined behavior of the burnt device.

PSID Field Structure

The PSID field is a 16-ascii (byte) character string. If the assigned PSID length is less then 16 characters, the remaining characters are filled with binary 0s by the burning tool.

The table below provides the format of a PSID.

Vendor Symbol	Board Type Symbol	Board Version Symbol	Parameter Set Number	Reserved
3 characters	3 characters	3 characters	4 characters	3 characters (filled with '\0')

Example:

A PSID for NVIDIA MHXL-CF128-T HCA board is MT_0030000001, where:

MT_ Vendor symbol

003 MHXL-CF128-T board symbol

000 Board version symbol

0001 Parameter Set Number

Assigning PSID and Integrating Flow

To assign and integrate the new PSID to produce the new FW:

1. Write the new FW configuration file (in .INI format).
2. Assign it with a PSID in the format described above. Use your own vendor symbol to ensure PSID uniqueness. If you do not know your vendor symbol, please contact your local NVIDIA FAE.
3. Set the PSID parameter in the new FW configuration file.

Remote Access to NVIDIA Devices

Burning a Switch In-band Device Using mlxburn

In order to update MT52000 Switch-IB device with a specific GUID (for example, 0xe41d2d03001094b0) using In-Band, the following steps are recommended:

> **Note**
>
> For Linux device names should be listed with the /dev/mst prefix. For Windows, no prefix is required.

1. Make sure all subnet ports are in the active state. One way to check this is to run opensm, the Subnet Manager.

```
[root@mymach]> /etc/init.d/opensmd start
opensm start [ OK ]
```

2. Make sure the local ports are active by running ‘ibv_devinfo’.

3. Obtain the device LID. There are two ways to obtain it:

   1. Using the “mst ib add” command:

      The “mst ib add”runs the ibdiagnet/ibnetdiscover tool to discover the InfiniBand fabric and then lists the discovered IB nodes as an mst device. These devices can be used for access by other MFT tools.

      ```
      [root@mymach]> mst ib add
      -I- Discovering the fabric - Running: /opt/bin/ibdiagnet
      -skip all
      -I- Added 3 in-band devices
      ```

   2. To list the discovered mst inband devices run “mst status”.

      ```
      devices[root@mymach]> mst status
      MST modules:
      ------------
      MST PCI module loaded
      MST PCI configuration module loaded
      ...
      Inband devices:
      -------------------
      /dev/mst/CA_MT4103_sw005_HCA-1_lid-0x0001
      /dev/mst/CA_MT4115_sw005_HCA-2_lid-0x0002
      /dev/mst/SW_MT52000_lid-0x0010
      [root@mymach]>
      ```

   3. Using the ibnetdiscover tool, run:

      ```
      ibnetdiscover | grep e41d2d03001094b0 | grep -w Switch
      Switch 36 "S-e41d2d03001094b0"
      ```

```
# "SwitchIB Mellanox Technologies" enhanced port 0 lid 16 lmc 0
```

Note

The resulting LID is given as a decimal number.

4. Run mlxburn with the LID retrieved in Step 3 above to perform the In-Band burning operation.

   Burn the Switch-INB device:

   ```
   # mlxburn -d lid-0x0010 -fw ./fw-SwitchIB.mlx
   -I- Querying device ...
   -I- Using auto detected configuration file: ./MSB7700-
   E_Ax.ini (PSID = MT_1870110032)
   -I- Generating image ...
   Current FW version on flash: 11.0.1250
   New FW version: 11.0200.0120
   Burning FS3 FW image without signatures - OK
   Restoring signature - OK
   -I- Image burn completed successfully.
   ```

In-Band Access to Multiple IB Subnets

In most cases, an adapter is connected to a single InfiniBand subnet. The LIDs (InfiniBand Local IDs) on this subnet are unique. In this state, the device access MADs are sent (to the target LID) from the first active port on the first adapter on the machine.

In case that the different IB ports are connected to different IB subnets, source IB port on the local host should be specified explicitly.

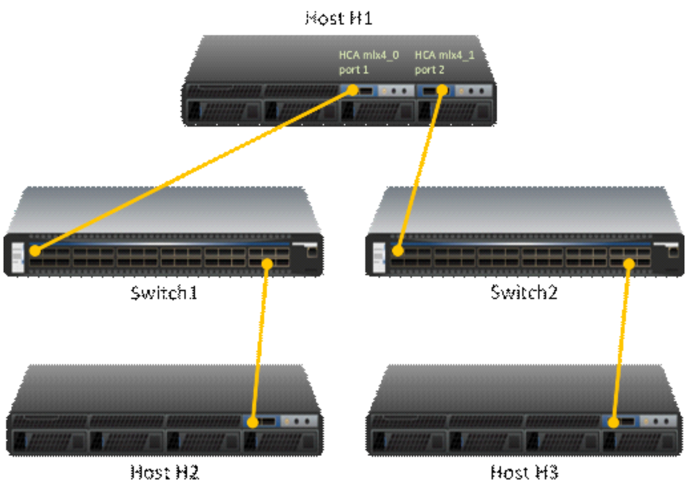
The device name would be in the format:

```
<any-string>lid-<lid-number>[,source adapter name][,source IB
port number]
```

For example:

- On Linux: lid-3,mlx4_0,1
- On Windows: lid-3,0,1

Say we have the following setup:



H1 host has 2 adapters. Port 1 of the first adapter is connected to Switch 1, and port 2 of the second adapter is connected to Switch 2. Since the 2 adapters on the H1 are not connected to the each other, there are 2 separate IB subnets in this setup.

Subnet1 nodes: H1 Switch 1 and H2 Subnet2 nodes: H1 Switch 2 and H3

Running "ibv_devinfo" command on H1 would list the 2 adapter names. For ConnectX adapters, the names would be mlx4_0 and mlx4_1.

Running "mst ib add" would add ib devices from the default port (first active port on the first adapter) - only Subnet1 nodes would be listed.

To add the nodes of the second subnet, the source adapter and port should be specified to the "mst ib add" command in the following format:

```
# mst ib add <hca_name> <hca_port>
```

Examples:

Add nodes of both subnets, Run:

```
# mst ib add mlx4_0 1
# mst ib add mlx4_1 2
```

List the devices:

```
# mst status
...
/dev/mst/CA_MT25418_H1_HCA-1_lid-0x0001,mlx4_0,1
/dev/mst/CA_MT25418_H2_HCA-1_lid-0x0005,mlx4_0,1
/dev/mst/SW_MT51000_Switch1_lid-0x0003,mlx4_0,1
/dev/mst/CA_MT25418_H1_HCA-1_lid-0x0010,mlx4_1,2
/dev/mst/CA_MT25418_H3_HCA-1_lid-0x0012,mlx4_1,2
/dev/mst/SW_MT51000_Switch2_lid-0x0005,mlx4_1,2
```

Note

You can use the above device names with the MFT tools.

MTUSB-1 USB to I2C Adapter

The MTUSB-1 is a USB to I2C bus adapter. This chapter provides the user with hardware and software installation instructions on machines running Linux or Windows operating systems.

MTUSB-1 Device

MTUSB-1 Package Contents

Please make sure that your package contains the items listed and that they are in good condition.

Item	Quantity	Description
MTUSB-1 device	1	USB to I2C bus adapter
USB cable	1	USB_A to USB_B (1.8m)
I2C cable	1	9-pin male-to-male cable (1.5m)
Converter cable	2	9-pin female to 3-pin (small/large) (0.3m)

System Requirements

The MTUSB-1 is a USB device which may be connected to any Personal Computer with a USB Host Adapter (USB Standard 1.1 or later) and having at least one USB connection port.

Supported Platforms

MTUSB-1 is supported in Linux and Windows only.

Hardware Installation

To install the MTUSB-1 hardware, please execute the following steps in the *exact* order:

1. Connect one end of the USB cable to the MTUSB-1 and the other end to the PC.

2. Connect one end of the I2C cable to the MTUSB-1 and the other end to the system/board you wish to control via the I2C interface. If the system/board uses a 3-pin connector instead of a 9-pin connector, connect the appropriate converter cable as an extension to the I2C cable on the 9-pin end, then connect its 3-pin end to the system/board.

Software Installation

The MTUSB-1 device requires that the MFT package be installed on the machine to which MTUSB-1 is connected; see MFT Installation for installation instructions.

For a Windows machine, it is also required to install the MTUSB-1 driver; visit http://www.diolan.com to download this driver. This driver is required for the first use of the MTUSB-1 device.

1. Start the *mst*1 driver. Enter: (Note: This step in not required in Windows.)

```
# mst start        (or mst restart if mst start was run
earlier)
```

2. To obtain the list of *mst* devices, enter:

```
# mst status -v       (or mst restart if mst start was run
earlier)
```

If MTUSB-1 has been correctly installed, **"mst status"** should include the following device in the device list it generates:

- On Linux: /dev/mst41:00.0/mtusb-1
- On Windows: mtusb-1

Switch Reprogramming through I2C Port

In order to reprogram the switch through the I2C adapter, follow the steps below:

For MSX1710/MSX67XX Switch systems:

1. Open the bus:

```
i2c -a 1 -d 1 /dev/mst/mtusb-1 w 0x60 0x20 0x10
i2c -a 1 -d 1 /dev/mst/mtusb-1 w 0x62 0x00 0x01
```

2. Burn the firmware:

```
flint -d /dev/mst/mtusb-1 -i ./fw-SX.bin b
```

3. Power cycle the system by unplugging and re-plugging the power cord to load the new firmware.

For MSX6025/6036 Switch systems:

1. Open the bus:

```
i2c -d /dev/mst/mtusb-1 w 0x22 0x1a 0xfb
```

2. Route the I2C bus to the switch device:

```
i2c -d /dev/mst/mtusb-1 w 0x70 0x0 0x1
```

3. Burn the firmware:

```
flint -d /dev/mst/mtusb-1 -i ./fw-SX.bin b
```

4. Power cycle the system by unplugging and re-plugging the power cord to load the new firmware.

Remote Access to Device by Sockets

The mst device on a machine can be accessed (server side) remotely for debugging purposes using the minimum set of tools from another machine (client side) which may have more tools or faster machine.

To do so:

- The mst server should run on the 'server side machine. Run: 'mst server start'
- The client side should add the mst 'server side'. Run: 'mst remote add <server side machine IP>'

After remote devices are added to the mst list device in the 'client side', you can run any tool that accesses the mst devices of the 'server side' as seen in the example below.

Usage of relevant command:

Command	Description
mst server start [port]	Starts mst server to allow incoming connection. Default port is 23108
mst server stop	Stops mst server.

Command	Description
mst remote add <host- name>[:port]	• Establishes connection with a specified host on a specified port (default port is 23108). • Adds devices on remote peer to local the devices list. • <hostname> may be host name as well as an IP address.
mst remote del <host- name>[:port]	Removes all remote devices on a specified hostname. <host-name>[:port] should be specified exactly as in the "mst remote add" command.

Example:

The example below shows how to query the firmware of a device in the server side (machine: mft) from the client side (machine: mft1):

1. Run mst status in the server side:

```
[root@mft ~]# mst status
MST modules:
------------
  MST PCI module loaded
  MST PCI configuration module loaded

MST devices:
------------
/dev/mst/mt4099_pciconf0     - PCI configuration cycles
access.
                               domain:bus:dev.fn=0000:0b:00.0
addr.reg=88 data.reg=92
                               Chip revision is: B0
/dev/mst/mt4099_pci_cr0      - PCI direct access.
                               domain:bus:dev.fn=0000:0b:00.0
bar=0xd2600000 size=0x100000
                               Chip revision is: B0
/dev/mst/mtusb-1:            - USB to I2C adapter as I2C
master
```

2. Start the mst server in the 'server side':

```
[root@mft ~]# mst server start
```

3. Add mst remote device in the client side:

```
[root@mft1 ~]# mst remote add mft
```

4. Show the mst device in the 'client side' which contains remote devices for the 'server side' machine:

```
[root@mft1 ~]# mst status
MST modules:
------------
  MST PCI module loaded
  MST PCI configuration module loaded

MST devices:
-----------
/dev/mst/mt4099_pciconf0       - PCI configuration cycles
access.
                                 domain:bus:dev.fn=0000:0b:00.0
addr.reg=88 data.reg=92
                                 Chip revision is: 01
/dev/mst/mt4099_pci_cr0        - PCI direct access.
                                 domain:bus:dev.fn=0000:0b:00.0
bar=0xd2600000 size=0x100000
                                 Chip revision is: 01
Remote MST devices:
-------------------
/dev/mst/mft:23108,@dev@mst@mt4099_pciconf0
                                 Chip revision is: B0
```

```
/dev/mst/mft:23108,@dev@mst@mt4099_pci_cr0
                                Chip revision is: B0
/dev/mst/mft:23108,@dev@mst@mtusb-1
```

5. Access a remote mst device from the 'client side':

```
[root@mft1 ~]# flint -d
/dev/mst/mft:23108,@dev@mst@mt4099_pci_cr0 q
Image type:       FS2
FW Version:       2.32.1092
FW Release Date:  17.8.2014
Rom Info:         type=PXE version=3.5.305 cpu=AMD64
Device ID:        4099
Description:      Node             Port1            Port2
Sys image
GUIDs:            0002c90300e6e4e0 0002c90300e6e4e1
0002c90300e6e4e2 0002c90300e6e4e3
MACs:                                  0002c9e6e4e1
0002c9e6e4e2
VSD:              n/a
PSID:             MT_1090120019
```

Accessing Remote InfiniBand Device by Direct Route MADs

To access IB devices remotely by direct route MADs (except for ConnectX-3 and ConnectX-3 Pro):

1. Make sure the local ports are connected to a node or more:

```
# ibstat
```

or

```
# ibv_devinfo
```

2. Obtain the device direct route path:

```
# mst ib add --use-ibdr --discover-tool ibnetdiscover mlx5_0
1
-I- Discovering the fabric - Running: ibnetdiscover -s -C
mlx5_0 -P 1
-I- Added 2 in-band devices
```

3. List the discovered direct route device:

```
# mst status MST modules:
  ------------
  MST PCI module loaded
  MST PCI configuration module loaded
MST devices:
------------
….
Inband devices:
-------------------
/dev/mst/CA_MT4113_server1_HCA-3_ibdr-0,mlx5_0,1
/dev/mst/SW_MT51000_switch1_ibdr-0.2,mlx5_0,1
```

4. Run any tool against the devices above.

```
#flint -d /dev/mst/CA_MT4113_server1_HCA-3_ibdr-0,mlx5_0,2 v
FS3 failsafe image
/0x00000038-0x00000f4f (0x000f18)/ (BOOT2) - OK
```

```
/0x00201000-0x0020101f (0x000020)/ (ITOC_Header) - OK
/0x00203000-0x0020323f (0x000240)/ (FW_MAIN_CFG) - OK
/0x00204000-0x0020437f (0x000380)/ (FW_BOOT_CFG) - OK
/0x00205000-0x002057ff (0x000800)/ (HW_MAIN_CFG) - OK
/0x00206000-0x002060ff (0x000100)/ (HW_BOOT_CFG) - OK
/0x00207000-0x002195e3 (0x0125e4)/ (PCI_CODE) - OK
/0x0021a000-0x0021e3a7 (0x0043a8)/ (IRON_PREP_CODE) - OK
/0x0021f000-0x00226bab (0x007bac)/ (PCIE_LINK_CODE) - OK
/0x00227000-0x002a888f (0x081890)/ (MAIN_CODE) - OK
/0x002a9000-0x002a95bf (0x0005c0)/ (POST_IRON_BOOT_CODE) - OK
/0x002aa000-0x002aa3ff (0x000400)/ (IMAGE_INFO) - OK
/0x002aa400-0x002b3e7b (0x009a7c)/ (FW_ADB) - OK
/0x002b3e7c-0x002b4277 (0x0003fc)/ (DBG_LOG_MAP) - OK
/0x002b4278-0x002b427f (0x000008)/ (DBG_FW_PARAMS) - OK
/0x003fa000-0x003fbfff (0x002000)/ (NV_DATA) - OK
/0x003fd000-0x003fd1ff (0x000200)/ (DEV_INFO) - OK
/0x003ff000-0x003ff13f (0x000140)/ (MFG_INFO) - OK
/0x003ff140-0x003ff13f (0x000000)/ (VPD_R0) - OK
FW image verification succeeded. Image is bootable.
```

Booting HCA Device in Livefish Mode

In case a MLNX HCA fails to boot properly, and is not being identified by the system due to a corrupt firmware, the user is able to boot the card in livefish mode, which allows re-burning of the flash device in order to restore the device into functional mode.

The device can also be forced into booting in livefish mode (only when supported on the board). To do so, a direct access to the card is needed. By connecting the two flash present pins using a jumper while the machine is powered off, the card will boot in "flash not present" mode (the firmware will not be loaded from the flash) i.e livefish.

Booting Card in Livefish Mode

To boot the card in livefish mode:

1. Power off the machine.

2. Locate the Flash preset pins on the HCA.
3. Close the two pins using a jumper.
4. Power on the machine.

Booting Card in Normal Mode

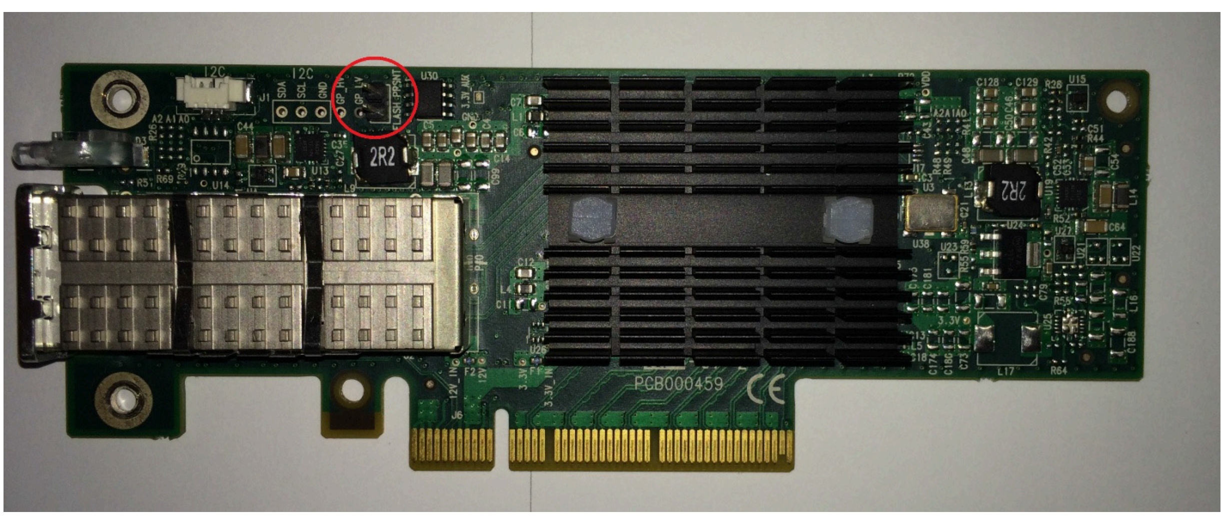
To boot the card in normal mode:

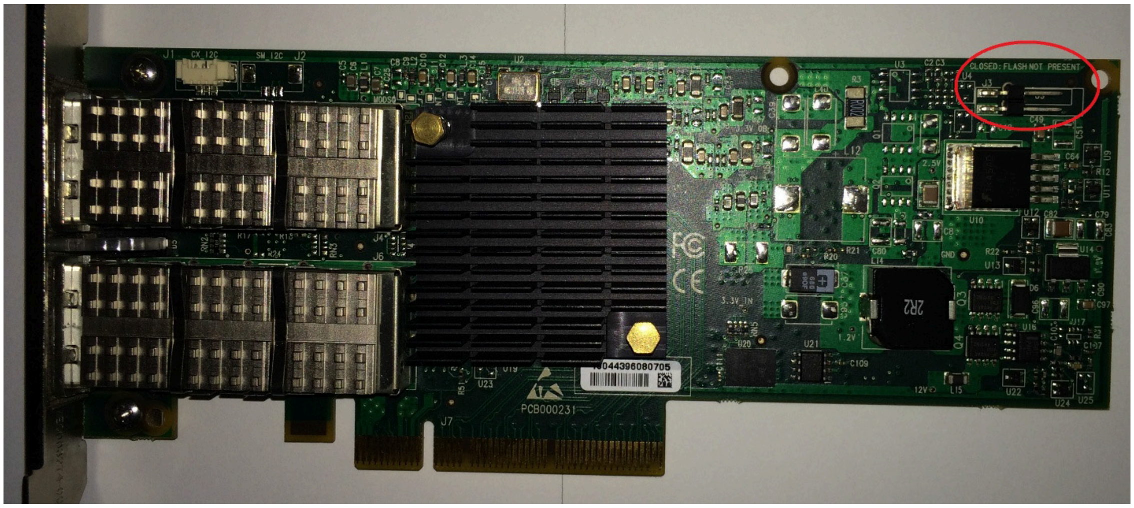
1. Power off the machine.
2. Take off the jumper connected to the Flash Present pins on the HCA.
3. Power on the machine.

Common Locations of Flash Present Pins

The following photos show common locations of the Flash Present pins.

> **Note**
>
> Existence and location of Flash Present pins depends on the board manufacture.





Burning a New Device

Select the appropriate method depending upon your device model.

- Connect-IB Adapter Card
- ConnectX-4 onwards Adapter Cards Family
- Spectrum or Switch-IB 2 Switch Systems
- Switch-IB Switch System

Connect-IB Adapter Card

When burning a flash for the first time, the initial image should contain the correct GUIDs and VPD for the device. Subsequent firmware updates will not change these initial setting.

> **Note**
>
> flint for OEM is required for burning Connect-IB for the first time.

GUIDs and MACs

The Connect-IB image contains the Node, Port and System GUIDs and Port MACs to be used by the card. To simplify GUIDs assignment, the mlxburn tool can derive the MACs and GUIDs from a single base GUID according to NVIDIA methodology:

```
Description: UID                  Number   Step
Port1 GUID:  base                 8        1
Port2 GUID:  base + 8             8        1
Port1 MAC:   guid2mac1(base)      8        1
Port2 MAC:   guid2mac(base + 8)   8        1
```

Note: guid2mac(guid) is (((guid >> 16) & 0xffffff000000) | (guid & 0xffffff)). Meaning, remove the 2 middle bytes of an 8 bytes GUID to generate a 6 bytes MAC.

PCI Vital Product Data (VPD)

The VPD information is returned by the firmware upon VPD access from the PCI configuration header.

- The vpd_ro file last 3 bytes are the vpd_rw tag-id and length
- The size of the vpd_r file (including the above 3 bytes) should be a multiple of 4

Burning a New Connect-IB Device

The VPD and GUIDs are stored in the last sector on flash that can be set as Write protected after the initial firmware burn.

Method 1: Generating Firmware with Specific GUIDs and Burning on the Flash

1. 1.Generate the initial image with the correct GUIDs and VPD for the specific device, using the mlxburn tool. The generated image occupies full flash size.

```
# mlxburn -fw FW/fw-ConnectIB.mlx -c FW/MCB194A-FCA_A1.ini -
wrimage fw-ConnectIB-MCB194A-FCA_A1.bin -base_guid
```

```
0x0002c903002ef500 -vpd_r_file ./vpd_r_data.bin
```

2. Disable the Write protection.

```
# flint -d /dev/mst/mt511_pciconf0 -ocr hw set
Flash0.WriteProtected=Disabled
```

3. Burn the entire flash, using the flint tool.

```
# flint -d /dev/mst/mt511_pciconf0 -i ./fw-ConnectIB-MCB194A-
FCA_A1.bin -ocr -ignore_dev_data -allow_psid_change -nofs --
yes burn
```

4. Set Write protection on the last sector, using mstflint:

   For devices using Winbond flash:

```
# flint -d /dev/mst/mt511_pciconf0 -ocr hw set
Flash0.WriteProtected=Top,1-SubSectors
```

5. Enable flash quad SPI IO operations.

```
# flint -d /dev/mst/mt511_pciconf0 -ocr hw set QuadEn=1
```

Method 2: Generating Firmware Image with Blank GUIDs, Burning and Setting GUIDs on the Flash

1. Generate the initial image with VPD for the specific device, using the mlxburn tool. The generated image occupies full flash size.

```
# mlxburn -fw FW/fw-ConnectIB.mlx -c FW/MCB194A-FCA_A1.ini -wrimage fw-ConnectIB-MCB194A-FCA_A1.bin -vpd_r_file ./vpd_r_data.bin
```

2. Disable the Write protection.

```
# flint -d /dev/mst/mt511_pciconf0 -ocr hw set Flash0.WriteProtected=Disabled
```

3. Burn the entire flash.

```
# flint -d /dev/mst/mt5111_pciconf0 -i ./fw-ConnectIB-MCB194A-FCA_A1.bin -ocr -ignore_dev_data -allow_psid_change -nofs --yes burn
```

4. Set device manufacture GUIDs.

```
# flint -d /dev/mst/mt511_pciconf0 -ocr -uid 0x0002c903002ef500 smg
```

5. Set device GUIDs.

```
# flint -d /dev/mst/mt511_pciconf0 -ocr -uid 0x0002c903002ef500 sg
```

6. Set Write Protection on the last sector, using the mstflint tool.

   For devices using Winbond flash:

   ```
   # flint -d /dev/mst/mt511_pciconf0 -ocr hw set
   Flash0.WriteProtected=Top,1-SubSectors
   ```

7. Enable flash quad SPI IO operations:

   ```
   # flint -d /dev/mst/mt511_pciconf0 -ocr hw set QuadEn=1
   ```

   1. To view flash settings, run:

      ```
      # flint -d /dev/mst/mt511_pciconf0 -ocr hw query
      ```

   2. To view assigned GUIDs, run:

      ```
      # flint -d /dev/mst/mt511_pciconf0 -ocr q
      ```

   3. To change a GUID after the initial burn, run:

      ```
      # flint -d /dev/mst/mt4113_pciconf0 -ocr -uid
      0x0002c903002ef500 sg
      ```

ConnectX-4 onwards Adapter Cards Family

Upon first time device burning, the GUIDs, MACs and VPD of the device are required to be set on the flash.

The sections below demonstrate two methods of burning a new ConnectX-4 onwards device in order to set these initial settings. Subsequent firmware updates will not change these settings.

Note

flint for OEM is required for burning ConnectX-4 onwards adapter cards family for the first time.

For information regarding GUIDs, MACs and VPD, please refer to Accessing Remote InfiniBand Device by Direct Route MADs.

Burning the ConnectX-4 onwards Adapter Cards Family

Method 1: Generating Firmware with Specific GUIDs and MACs and Burning it on the Device

In order to burn a new device, follow the steps below:

1. Disable the Write protection.

```
# flint -d /dev/mst/mt521_pciconf0 -ocr hw set
Flash0.WriteProtected=Disabled
```

2. Burn the entire flash.

```
# flint -d /dev/mst/mt521_pciconf0 -i ./ fw-ConnectX4-
MCX454_Ax.bin -ocr -ignore_dev_data - allow_psid_change -nofs
--yes burn
```

3. Set Write protection

```
# flint -d /dev/mst/mt521_pciconf0 -ocr hw set
Flash0.WriteProtected=Top,8-SubSectors
```

4. Enable flash quad SPI IO operations.

```
# flint -d /dev/mst/mt521_pciconf0 -ocr hw set QuadEn=1
```

Method 2: Generating Firmware Image with Blank GUIDs, Burning and Setting GUIDs on the Device

In order to burn a new device, follow the steps below:

1. Disable the Write protection.

```
# flint -d /dev/mst/mt521_pciconf0 -ocr hw set
Flash0.WriteProtected=Disabled
```

2. Burn the entire flash.

```
# flint -d /dev/mst/mt521_pciconf0 -i ./ fw- ConnectX4-
MCX454_Ax.bin -ocr -ignore_dev_data - allow_psid_change -nofs
```

```
--yes burn
```

3. Set device manufacture GUIDs and MACs.

```
# flint -d /dev/mst/mt521_pciconf0 -ocr -guid 0xe41d2d0300570fc0
-mac 0x0000e41d2d570fc0 smg
```

4. Set device GUIDs and MACs.

```
 # flint -d /dev/mst/mt521_pciconf0 -ocr -guid 0xe41d2d0300570fc0
-mac 0x0000e41d2d570fc0 sg
```

5. Set Write protection on the last sector using flint:

```
# flint -d /dev/mst/mt521_pciconf0 -ocr hw set
Flash0.WriteProtected=Top,8-SubSectors
```

Note

The command may vary based on the Flash type used.

Protection_type can be :

- 1- Top,8-SubSectors, if it is in [W25QxxBV]
- 2- Top,1-Sectors, if it is in [MX25L16xxx, N25Q0XX, IS25LPxxx, S25FL256L, MX25Lxxx, W25Qxxx, MX25Uxxx]

6. Enable flash quad SPI IO operations.

```
# flint -d /dev/mst/mt521_pciconf0 -ocr hw set QuadEn=1
```

1. To view flash settings:

```
# flint -d /dev/mst/mt521_pciconf0 -ocr hw query
```

2. To view assigned GUIDs:

```
# flint -d /dev/mst/mt521_pciconf0 -ocr
```

3. To change a GUID after the initial burn:

```
# flint -d /dev/mst/mt4115_pciconf0 -ocr -guid
0xe41d2d0300570fc0 sg
```

4. To change a MAC after the initial burn:

```
# flint -d /dev/mst/mt4115_pciconf0 -ocr -mac
0x0000e41d2d570fc0 sg
```

5. To change a GUID and derive MAC from it after the initial burn, run:

```
# flint -d /dev/mst/mt4115_pciconf0 -ocr -uid
0xe41d2d0300570fc0 sg
```

Spectrum or Switch-IB 2 Switch Systems

Upon first time flash burning, the GUIDs and VPD of the device are required to be set on the flash. The sections below demonstrate two methods of burning a new device in order to set these initial settings. Subsequent firmware updates will not change these settings.

> **Note**
>
> flint for OEM is required for burning Spectrum/Switch-IB 2 for the first time.

For information regarding GUIDs, MACs and VPD, please refer to Accessing Remote InfiniBand Device by Direct Route MADs.

Burning the Spectrum/Switch-IB 2 Device

Method 1: Generating Firmware with Specific GUIDs and MACs and Burning it on Device

In order to burn a new Spectrum/Switch-IB 2 device, follow the steps below:

1. Generate the initial image with the correct GUIDs and VPD for the specific device using the mlxburn tool. The generated image occupies full flash size.

```
# mlxburn -fw FW/fw-Spectrum.mlx -c FW/MCB194A-FCA_A1.ini -
wrimage fw-Spectrum-MCB194A- FCA_A1.bin -base_guid
0x0002c903002ef500 -base_mac 0x02c90ef500 -vpd_r_file
./vpd_r_data.bin
```

2. Disable Write protection.

```
# flint -d /dev/mst/mt585_pciconf0 -
override_cache_replacement hw set Flash0.WritePro-
tected=Disabled
```

3. Burn the entire flash.

```
# flint -d /dev/mst/mt585_pciconf0 -i ./fw-Spectrum-MCB194A-
FCA_A1.bin -override_cache_re- placement -ignore_dev_data -
nofs -allow_psid_change -y b
```

4. Set Write protection.

```
# flint -d /dev/mst/mt585_pciconf0 -
override_cache_replacement hw set
Flash0.WriteProtected=Top,8-SubSectors
```

5. Enable flash quad SPI IO operations.

```
# flint -d /dev/mst/mt585_pciconf0 -
override_cache_replacement hw set QuadEn=1
```

Method 2: Generating a Firmware Image with Blank GUIDs, Burning and Setting GUIDs on the Device

In order to burn a new Spectrum/Switch-IB 2 device, follow the steps below:

1. Generate the initial image VPD for the specific device using the mlxburn tool. The generated image occupies full flash size.

```
# mlxburn -fw FW/fw-Spectrum.mlx -c FW/MCB194A-FCA_A1.ini -
wrimage fw-Spectrum-MCB194A- FCA_A1.bin -vpd_r_file
./vpd_r_data.bin
```

2. Disable Write protection.

```
# flint -d /dev/mst/mt585_pciconf0 -
override_cache_replacement hw set Flash0.WritePro-
tected=Disabled
```

3. Burn the entire flash using the flint tool.

```
# flint -d /dev/mst/mt585_pciconf0 -i ./fw-Spectrum-MCB194A-
FCA_A1.bin -ocr -ignore_dev_data -nofs -allow_psid_change -y
b
```

4. Set device manufacture GUIDs and MACs.

```
# flint -d /dev/mst/mt585_pciconf0 -ocr -guid 0xe41d2d0300570fc0
-mac 0x0000e41d2d570fc0 smg
```

5. Set device GUIDs and MACs.

```
# flint -d /dev/mst/mt585_pciconf0 -ocr -guid 0xe41d2d0300570fc0
-mac 0x0000e41d2d570fc0 sg
```

6. Set Write protection on the last sector.

```
# flint -d /dev/mst/mt585_pciconf0 -ocr hw set
Flash0.WriteProtected=Top,8-SubSectors
```

7. Enable flash quad SPI IO operations.

```
# flint -d /dev/mst/mt585_pciconf0 -ocr hw set QuadEn=1
```

1. To view flash settings:

```
# flint -d /dev/mst/mt53000_pciconf0 -ocr hw query
```

2. To view assigned GUIDs:

```
# flint -d /dev/mst/mt53000_pciconf0 -ocr q
```

3. To change a GUID after the initial burn:

```
# flint -d /dev/mst/mt53000_pciconf0 -ocr -guid
0xe41d2d0300570fc0 sg
```

4. To change a MAC after the initial burn:

```
# flint -d /dev/mst/mt53000_pciconf0 -ocr -mac
0x0000e41d2d570fc0 sg
```

5. To change a GUID and derive MAC from it after the initial burn, run:

```
# flint -d /dev/mst/mt53000_pciconf0 -ocr -uid
0xe41d2d0300570fc0 sg
```

Switch-IB Switch System

Upon first time flash burning, the GUIDs and VPD of the device are required to be set on the flash.

The sections below demonstrate two methods of burning a new Switch-IB device in order to set these initial settings. Subsequent firmware updates will not change these settings.

Note

flint for OEM is required for burning Switch-IB for the first time.

For information regarding GUIDs, MACs and VPD, please refer to Accessing Remote InfiniBand Device by Direct Route MADs.

Burning the Switch-IB Device

The examples below are for managed switches. For unmanaged switches, connect an MTUSB adapter (see MTUSB-1 USB to I2C Adapter) to the device and use the appropriate mst device (/dev/mst/mtusb...).

Method 1: Generating Firmware with Specific GUIDs and Burning it on the Flash

In order to burn a new Switch-IB device, follow the steps below:

1. Generate the initial image with the correct GUIDs and VPD for the specific device using the mlxburn tool. The generated image occupies full flash size.

```
# mlxburn -fw FW/fw-SwitchIB.mlx -c FW/MSB7700-E_Ax.ini -
wrimage fw-SwitchIB-MSB7700-E_Ax.bin -base_guid
0x0002c903002ef500 -vpd_r_file ./vpd_r_data.bin
```

2. Disable the Write protection.

```
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set
Flash0.WriteProtected=Disabled
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set
Flash1.WriteProtected=Disabled
```

3. Burn the entire flash.

```
# flint -d /dev/mst/mt583_pciconf0 -i ./ fw-SwitchIB-MSB7700-
E_Ax.bin - ocr -ignore_dev_data -allow_psid_change -nofs --
yes burn
```

4. Set Write protection.

```
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set
Flash0.WriteProtected=Top,2-SubSectors
```

5. Enable flash quad SPI IO operations.

```
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set QuadEn=1
```

Method 2: Generating Firmware Image with Blank GUIDs, Burning and Setting GUIDs on the Flash

In order to burn a new Switch-IB device, follow the steps below:

1. Generate the initial image VPD for the specific device using the mlxburn tool. The generated image occupies full flash size.

```
# mlxburn -fw FW/fw-SwitchIB.mlx -c FW/MSB7700-E_Ax.ini -
wrimage fw-SwitchIB-MSB7700-E_Ax.bin
-vpd_r_file ./vpd_r_data.bin
```

2. Disable the Write protection.

```
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set
Flash0.WritePro-tected=Disabled
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set
Flash1.WritePro-tected=Disabled
```

3. Burn the entire flash.

```
# flint -d /dev/mst/mt583_pciconf0 -i ./ fw-SwitchIB-MSB7700-
E_Ax.bin - ocr -ignore_dev_data -allow_psid_change -nofs --
yes burn
```

4. Set device manufacture GUIDs.

```
# flint -d /dev/mst/mt583_pciconf0 -ocr -uid 0x0002c903002ef500
smg
```

5. Set device GUIDs.

```
# flint -d /dev/mst/mt583_pciconf0 -ocr -uid 0x0002c903002ef500
sg
```

6. Set Write protection.

```
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set
Flash0.WriteProtected=Top,2-SubSectors
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set
Flash1.WriteProtected=Top,1-SubSectors
```

7. Enable flash quad SPI IO operations.

```
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set QuadEn=1
```

    1. To view flash settings:

    ```
    # flint -d /dev/mst/mt583_pciconf0 -ocr hw query
    ```

    2. To view assigned GUIDs:

    ```
    # flint -d /dev/mst/mt583_pciconf0 -ocr q
    ```

    3. To change a GUID after the initial burn:

```
# flint -d /dev/mst/mt52000_pciconf0 -ocr -uid 0x0002c903002ef500 sg
```

Generating Firmware Secure and NV LifeCycle Configurations Files

Create Forbidden Versions Binary File

The flint "set_forbidden_versions" command takes as a parameter the binary file that contains the forbidden versions. To create this file easily you can use the mlxconfig "xml2bin" command. The forbidden versions configuration is found in the mlxconfig database.

1. Run mlxconfig "gen_tlvs_file".
2. Choose "nv_forbidden_versions".
3. Generate XML template using mlxconfig "gen_xml_template".
4. Set values for the parameters in the XML template.

   You can set up to 32 forbidden versions. mlxconfig requires all the parameters in the XML template to have values, so in case you want to fill only one forbidden version you have to set the other 31 parameters to zero, you can do that by using index range as follows:

   ```
   <forbidden_fw_version index='1..31' >0x0</forbidden_fw_version>
   ```

5. Run the xml2bin command to generate the binary file.

Create Tokens for Secure Firmware and NV LifeCycle

mlxconfig can be used to generate CS tokens, debug tokens and NV LifeCycle configuration files and apply it on your device.

To create the tokens:

1. Generate XML template that contains the necessary configurations for the token.

   The XML template must have only one token configuration. The current available token configurations are: debug token, challenge based debug token, customer token, challenge based customer token and MLNX ID token.

   The XML must also have the `file_applicable_to` configuration for all devices and an additional configuration which is device dependent:

   1. For cables include the `file_device_unique` configuration.
   2. For other devices include the `file_mac_addr_list` configuration.

2. Generate a binary file that can be applied to the device using the mlxconfig create_conf command.

Updating Firmware Using ethtool/devlink and .mfa2 File

In order to flash the firmware on the device using ethtool, you need to prepare a .mfa2 firmware file using the mlxarchive tool - see mlxarchive – Binary Files Compression Tool. Note that mlxarchive requires installing MFT with `--oem` option.

To perform firmware upgrade using ethtool/devlink, follow the steps below:

images/download/thumbnails/4291540909/Procedure_Heading_Icon-version-1-modificationdate-1757922508740-api-v2.PNG

1. Run the mlxarchive tool to generate the .mfa2 file (the following example assumes MFA2 v1.1.1).

```
# mlxarchive -v 1.1.1 --bins-dir <source binaries directory> -
-out-file /lib/firmware/<file_name>.mfa2
```

2. Obtain the interface name of the adapter for which you wish to update firmware. For example, you can use `ifconfig -a`.

```
# ifconfig -a
...
p5p1: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether ec:0d:9a:48:af:2a  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0
collisions 0

p5p2: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether ec:0d:9a:48:af:2b  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0
collisions 0
...
```

3. Burn the firmware using the .mfa2 image with ethtool/devlink. Please use the .mfa2 file path relative to /lib/firmware.

ethtool command:

```
# ethtool -f <interface name> <file_name>.mfa2
```

devlink command:

```
$ devlink dev flash <dev> file <file_name>.mfa2
```

4. Query the adapter to verify that the new firmware version has been loaded following.

```
# mst start
# mst status -v
MST modules:
------------
    MST PCI module loaded
    MST PCI configuration module loaded
PCI devices:
------------
DEVICE_TYPE             MST                            PCI
RDMA            NET                         NUMA
ConnectX5(rev:0)        /dev/mst/mt4119_pciconf0.1     05:00.1
mlx5_3          net-p5p2                    0
ConnectX5(rev:0)        /dev/mst/mt4119_pciconf0       05:00.0
mlx5_2          net-p5p1                    0
...
#

# flint -d /dev/mst/mt4119_pciconf0 q
Image type:             FS4FW Version: 16.25.1042 FW
Version(Running): 16.25.1020 FW Release Date: 30.4.2019 Product
Version: 16.25.1020 Rom Info: type=UEFI version=14.18.19
cpu=AMD64 type=PXE version=3.5.701 cpu=AMD64 Description: UID
GuidsNumber Base GUID: ec0d9a030048af2a 4 Base MAC:
ec0d9a48af2a 4 Image VSD: N/A Device VSD: N/A PSID:
MT_0000000080 Security Attributes: N/A
```

5. For the firmware update to take effect, you need to either reboot the server or run:

```
# mlxfwreset -d /dev/mst/mt4119_pciconf0 -y r
```

6. Validate the firmware update by a query.

Using mst:

```
# mst start

# flint -d /dev/mst/mt4119_pciconf0 q
Image type:             FS4
FW Version:             16.25.1042
FW Release Date:        15.5.2019
Product Version:        16.25.1042
Rom Info:               type=UEFI version=14.18.19 cpu=AMD64
                        type=PXE version=3.5.701 cpu=AMD64
Description:            UID                GuidsNumber
Base GUID:              ec0d9a030048af2a        4
Base MAC:               ec0d9a48af2a            4
Image VSD:              N/A
Device VSD:             N/A
PSID:                   MT_0000000080
Security Attributes:    N/A
#
```

Using devlink:

```
$ devlink dev info <dev>
pci/0000:05:00.0:
  driver mlx5_core
  versions:
      fixed: fw.psid MT_0000000080
      running: fw.version 16.23.1000
```

```
        stored: fw.version 16.25.1042
```

Using MFT tools on Secured GPU Devices

In systems with secured FW, GPUs cannot be accessed directly via PCIe, therefore "pci_cr" devices will become inaccessible.

The only way to send firmware queries is via the NVIDIA GPU Driver.

Please follow these instructions to discover and display GPU devices on your secured system:

1. Make sure the NVIDIA driver is loaded using "modprobe nvidia".
2. Start the mst service by running "mst start" command.
3. Discover GPU devices via the NVIDIA Driver using the "mst gpu add" command.
4. See the newly added devices using "mst status -v" command, they will appear under "GPU Devices".

Run any of the supported tools using the GPU NETIR mst device. For example:

`mlxlink -d /dev/mst/netir10561_08.1.00_gpu0`

ESXi

The MFT and MFT-OEM packages in VMware now include a new plugin feature. This plugin can be accessed and managed using the following command within the ESXi shell:

esxcli mellanox <package name> <tool name> <command>

For example:

esxcli mellanox mst status

To enable this functionality without requiring additional code for each tool, a wrapper has been implemented. This wrapper encapsulates the tool's command and ensures the output is returned in the correct XML format.

Additionally, a plugin generator has been developed. This generator automatically creates the necessary XML files for each tool during the packaging process, streamlining the integration of new tools into the plugin framework.

There are a few challenges with integrating existing tools into the ESXi plugin:

- Tools that generate output during their execution (e.g., progress indicators like percentage bars in mlxfwmanager) can cause issues with the VMware plugin if not handled properly
    - To address this, some tools might need to provide an option to suppress such progress indicators. For example, mlxfwmanager includes the `--no-progress` flag for this purpose
- If a tool requires user input or prompts during its execution, it must provide clear mechanisms for handling these interactions within the context of the VMware plugin
    - This may include options for non-interactive execution (e.g., `--yes`, `--force`) or clear and well-defined prompts with expected input formats and error handling
- If a tool prints output with colors, the wrapper converts the color output to regular, otherwise the VMware plugin will fail

In summary:

- Each tool might require specific adjustments or workarounds to ensure compatibility with the VMware plugin.
- In some cases, the tool's output format or behavior may be inherently incompatible with the plugin's requirements, preventing successful integration.

By carefully considering these factors and implementing appropriate solutions (such as using suppression flags or modifying tool behavior), we can maximize the number of tools that can be successfully integrated with the VMware plugin.

Supported plugin commands:

Description	Original Command	esxcli plugin command
Create special files that represent Nvidia devices in directory /dev/mst. After successfully completion of this command the MST driver is ready to work and you can invoke other Nvidia tools.	/opt/mellanox/bin/**mst start**	esxcli mellanox mft **mst start**
Stop Nvidia MST driver service.	/opt/mellanox/bin/**mst stop**	esxcli mellanox mft **mst stop**
Show all Nvidia devices connected to server	/opt/mellanox/bin/**mst status**	esxcli mellanox mft **mst status**
Show all Nvidia devices connected to server [verbose mode]	/opt/mellanox/bin/**mst status -v**	esxcli mellanox mft **mst status -v** esxcli mellanox mft **mst status --verbose**
Query all devices connected to server	/opt/mellanox/bin/**mlxfwmanager**	esxcli mellanox mft **mlxfwmanager query**
Query device and look for available online FW for the given device from NVIDIA web-site , if current FW version older than latest one , FW upgrade will be done	/opt/mellanox/bin/**mlxfwmanager --online —-update --no-progress --yes**	esxcli mellanox mft **mlxfwmanager update**
Burn FW image to a given device	/opt/mellanox/bin/**flint -d <dev> -i <image> --silent -y b**	esxcli mellanox mft **flint burn -d <dev> -i <image> --run**
start MST server for remote connections	/opt/mellanox/**mst server start -s <pass>**	esxcli mellanox mft **mst serverStart -s <passphrase> -p <port>**
Stop MST server	/opt/mellanox/**mst server stop**	esxcli mellanox mft **mst serverStop**

Query specific device connected to server	/opt/mellanox/bin/**flint -d <dev> q**	esxcli mellanox mft **flint query -d <dev> --run**
Query specific device connected to server with full option	/opt/mellanox/bin/**flint -d <dev> q full**	esxcli mellanox mft **flint query -d <dev> --run --full**
Verify entire device flash	/opt/mellanox/bin/**flint -d <dev> v**	esxcli mellanox mft **flint verify -d <dev> --run**
Show all mlxreg available supported registers for the given device	/opt/mellanox/bin/**mlxreg -d <dev> show_regs**	esxcli mellanox mft **mlxreg showRegs -d <dev>**
The dump file is used by the Support team for hardware troubleshooting purposes. It can be applied on all NVIDIA devices.	/opt/mellanox/bin/**mstdump <dev> device_name.dmp**	esxcli mellanox mft **mstdump dump -d <dev>**
Read the hardware temperature from NVIDIA devices with temperature sensors (all NVIDIA devices) and prints the result in Celsius degrees.	/opt/mellanox/bin/**mget_temp -d <dev>**	esxcli mellanox mft **mgettemp read -d <dev>**
Show NVIDIA device link status	/opt/mellanox/bin/**mlxlink -d <dev>**	esxcli mellanox mft **mlxlink status -d <dev>**

ⓘ **Note**

If you're not sure how to run a command or you want to explore the available flag shortcuts in the esxcli plugin, you can check it by writing a partial start of a command and utilizing the output you're getting to figure out what to write next.

Example: flint tool

Write the beginning of the command like this:

esxcli mellanox mft flint

the output will be the available options for the continuation of the command:

```
[root@apps-35:~] esxcli mellanox mft flint
Usage: esxcli mellanox mft flint {cmd} [cmd options]

Available Commands:
  burn                Burn flash.
  query               Query misc. flash/firmware characteristics, use "full" to get more information.
  verify              Verify entire flash.
  version             Display flint version
[root@apps-35:~]
```

And you can continue the command with the option you'd like to proceed with, to see which flags are needed for the chosen command.

For example, if you'd like to choose the "verify" command, write:

esxcli mellanox mft flint verify

the following output will be dispalyed:

```
[root@apps-35:~] esxcli mellanox mft flint verify
Error: Missing required parameter -r|--run

Usage: esxcli mellanox mft flint verify [cmd options]

Description:
  verify                Verify entire flash.

Cmd options:
  -d|--device=<str>     Device flash is connected to.
  -r|--run              Use "--run" to execute command. (required)
[root@apps-35:~]
```

Please note that in the flint command the user must use the --run or -r flag to run the command.

If the user forgets to use the required command, an informative error message will be displayed:

```
[root@apps-35:~] esxcli mellanox mft flint verify -d mt4119_pciconf0
Error: Missing required parameter -r|--run

Usage: esxcli mellanox mft flint verify [cmd options]

Description:
  verify                Verify entire flash.

Cmd options:
  -d|--device=<str>     Device flash is connected to.
  -r|--run              Use "--run" to execute command. (required)
[root@apps-35:~]
```

Document Revision History

Release Notes Revision History

MFT Release Notes Change Log History

Component/ Tool	Description
Rev. 4.33.0-169	
mlxfwmanager	Added the ability to skip firmware update when the target is already burned on the device and the user used the skip_if_same flag. Usage: • `mlxfwmanager -d <> -u --skip_if_same -i <.mfa>` • `flint -d <> -i <.bin> --skip_if_same b`
flint	Added support for flint to query firmware images containe within PLDM (.fwpkg) package files.
mlxlink	Added support for a new field for --show_module flag.
	Added support for a new capability to allow the user to configure recovery types using the flag --phy_recovery. The recovery types available are: • host_serdes_feq • host_logic_re_lock
mlxfwreset	Added support for sync 2 to reset a PCIe switch using a hc reset using the arguments --sync 2 and --method 1.
Rev. 4.32.0-120	

Component/ Tool	Description
mlxptrace	Added support for --raw_dump flag for mlxptrace utility. Th flag allows users to take a raw resource_dump of HW_TRA(segment and translate it to readable traces, as if they ware taken from mlxptrace.
flint	Added support for displaying device mode by adding eithe independent, standalone or unknown mode to the flint har query output.
	Added support for displaying TB/BP bits as-is and for displ the SRWD bit for specific flash types.
	Added support for Independent module Stand Alone optio SA mode on Moose is enabled.
mlxlink	Added support for FEC correctable and FEC uncorrectable in mlxlink --port_type PCIE -c.
	Added support for aggregating module information for all ASICs for an aggregated port. This features enables to pre single value for each field instead of presenting for each li separately.
	Added support for mlxlink port mapping to FNM.
	Added support for -m options in mlxlink.
	Added new fields to support the ELS and OE unique characteristics and required telemetry.
	Added support for a new flag "module_state" with three st (up, down, tg) to allow the user to change the module state
	Added a feature to PRBS to support dc_cpl_allow flag to ru PRBS on DC coupled links. Getting a DC coupled ports into test\PRBS mode will require the user's approval, to avoid H shortage.
	Added support to display a detailed BER count for each lar
mlxconfig	Rearranged the way BYTE type TLVS are handled in mlxcon properly support customization_number.
	Added support to output configuration to file as JSON.
mlxreg	Removed PRM reg index enforcement from mlxreg tool.
mlxfwmanager	Added support for flashing firmware binary image stored i

Component/ Tool	Description
	PLDM package for flint and mlxfwmanager tools.
mlxptrace	Added support for --raw_dump flag for mlxptrace tool. This feature supports hardware Tracer Dump From Resource Du
Rev. 4.31.0-149	
flint	SECURITY_LOG section was added to image layout. Added FS5_SECURITY_LOG identifier which will be displayed durin command: `flint -i <bin> verify`
	flint -d <dev> -ocr hw query: Added support for querying B (Block Protection) and TBS (Top Bottom Selection) in raw, b form. flint -d <dev> -ocr hw set: Added support for performing p sets for write protection. This allows users to set either the TBS bit or the BP bits individually, instead of requiring both to be set simultaneou
	Added support for displaying flint query if the device is Soc Direct device (If not - nothing new will be printed). For BF-3 devices, will also display if aux card is connected.
	Added support for querying if the Aux card is connected by of MRSV.two_p_core_active. This is meaningful only if the case where the two_p_core_a strap is fed from the AUX card plug.
mlxdpa	Added support for host ELF file to appear in a format wher each DPA app can contain multiple DPA ELFs for different hardware arch versions. Command: `mlxdpa -e <host elf file> -c <certificate pat p <private key path> -o <output path> sign_dpa_apps`
mst	Added support for creating a planarized mst device file by providing a SystemImageGUID and the 4 lids that share it. Command: `mst ib create_aggregated_device --system_image_guid <> --lids <>` In case the IB devices were created with mlx5_1, the comm should be as follows:

Component/ Tool	Description
	`mst ib create_aggregated_device --system_image_guid <> --lids <> --hca_id mlx5_`
mlxconfig, mlxtokengenerator	mlxconfig and mlxtokengenerator support ConnextX-7 anc BlueField-3 devices with challenge-based tokens.

mlxconfig	Added support to output configuration to file as JSON.
	Command:
	`mlxconfig -d /dev/mst/mt4129_pciconf0 -j /tmp/configuration.json query`
	`mlxconfig -d mt4129_pciconf0 --json_format C:\Users\Administrator\Desktop\configuration. -e query NUM_OF_VFS`

mget_temp	Added support for get temperature tool to mstflint.
mlxreg	The tool no longer enforces indexes, indexes that are not specified will be set to zero.
General	Added support for RDMA devices modified names.
	Added support to detect ConnectX location using straps (t allow Same INI for both ConnectX).
Rev. 4.30.0	
flint	Detect ConnectX location using straps. If a device is multi / system component, display its geographical address to the Example (for a ASIC platform device): Geographical Address: ASIC 3 Note that for a device that is not multi ASIC system compc no such data will be displayed.
	Added the option to enable the user to query the certificat status after burning the DPA cert. This tool will print a list c the certificates and their metadata. If the command includes cert uidd, the certificate will be w to the provided file. Command: `flint -d <device_name> --component_type digital_cacert [--cert_uuid <cert uuid>] query_components [output file]`
mlxreg	Added the option `--overwrite` for sending a SET comm without sending an initial GET command that fills the

Component/ Tool	Description
	unspecified non-index fields. Example: `mlxreg -d <device_name> --reg_name MNVIA --overwrite --set "writer_id=9"`
resourceparse/resourcedump	Added the option `--out-dir` to write each segment into different file under the supplied directory. Examples: • `resourcedump dump -d <device_name> -s <segment> --out-dir <output_directory>` • `resourceparse -d <dump_file> -p map --ou dir <output_directory>`
	Added the option `--hide-segment-header` to hide seg header from ADB parse output.
General	Implemented new reset flow for PCIe Switch and Connect- devices to reset the device using HOT reset mechanism.
	Added the option to allow users to burn the QM3 FW on th BMC server using the Redfish interface.
	Added support for recovering a device from Zombiefish mo device in Zombiefish mode exposes the functional device I it looks like a functional device, though it does not have val firmware running on it). To recover a device from Zombiefish mode, run the followir command: `flint -ocr -d -i -y b`
Rev. 4.29.0	
General	Added a warning message before deprecating the support OpenSSL engine in the next release.
flint	Added support to print new INI version field.
mlxtokengenerator	Added support for challenge data from blob file instead of device. Example of use: "mlxtokengenerator -b <challenge_blob.bir CRCS -t switch -o /tmp/crcs_token.xml generate_token
Rev. 4.28	

Component/ Tool	Description
General	Added support for flint query to display the port GUID, noc GUID, system GUID, and allocated GUID for devices that are of a multi-asic system.
	Expanded the MCC register to include more informative er messages which are relevant for module upgrade flows.
	Upgraded full OS.
	Added support for new flag "-s" which allows the user to determine which I2C secondary address to use when runni mget_temp tool
	Added an error message stating that the image/device is encrypting and breaking the flow.
	Added support in mlxtrace and fwtrace tools to read event ArcusE.
mlxlink	Disabled physical state in optic cables.
mlxfwreset	The mlxfwreset tool might fail when using PPC64LE on the 8.7 operating system.
	Changed the default firmware reset to level 3 from level 4 PCIe devices. To load new configurations, either execute mlxfwreset leve initiate a cold boot.
Rev. 4.27.0-83	
General	When using the TACACS protocol to check scripts that are run during login, the MFT auto-completion generation is activated. The auto-complete files are now available in a se package, which is an optional installation.
	Added flint support for querying and upgrading firmware c modules of part number MMA4Z00-NS.
mlxlink	Added mlxlink support for read/write/dump actions via dire EEPROM. This is a Beta level support feature that is curren limited to IEI file-systems.
Rev. 4.26.1	
Bug Fixes	See MFT Bug Fixes History.
Rev. 4.26	

Component/ Tool	Description
mlxconfig	Removed the `--read_only` flag from the mlxconfig tool.
General	Added support for an additional flash type - Winbond Part W25Q256JVFIQ.
General	Removed dependency on Boost library.
General	Added support for VPATH builds.
General	The descriptors for DEB/RPM packages are now configured include two additional tools: mstreg and mstlink. This resul the addition of build-time dependencies on `libexpat1-d` and `liblzma-dev`.
Rev. 4.25.1	
mlxfwreset	Added a query for the last reboot. The "`mlxfwreset -d` command now shows the cause for the last reboot, and th number of clock cycles since the last cold reset.
mlxconfig	Added a priority field to the XML header of each TLV in the mlxconfig xml file. The priority possible values are "MLNX", and "USER".
Rev. 4.25.0	
General	Added MFT package for WinPE running on Arm64 (aarch64 processors.
	Added Sha256 signature to MFT RPM packages in order tc installation on FIPS (Federal Information Processing Standa enabled systems.
mlxconfig	Added --read_only as a new flag to the mlxconfig tool. Whe query with this flag is enabled, the user is able to see read_ paramters. These parameters are marked with 'RO' in the c
mlxfwreset	Added support for DGX H100 device reset. All DGX H100 d reset requests are handled by the mlxfwreset tool. When o the DGX H100 devices is provided, the tool proceeds to res the devices accordingly. As the final step in the reset proce reboot command is executed, resulting in a reboot of the e setup.
mlxlink	Added SNR (signal-noise ratio) for the media and host side active\optical NDR modules.

Component/ Tool	Description
mlxdpa	mlxdpa was added the ability to create and sign containers the addition and removal of certificates.
mlxdump	Starting from this release, the fsdump mode of the mlxdur tool is deprecated. Support will be provided by November 2 Please use resource-dump with the appropriate segment ir
Rev. 4.24.0	
mlxlink	Updated the `show_module` command output to display R TX power in a higher resolution.
	Updated the `show_module` command output to display n module information fields - Round Trip Latency, Intra-ASIC Latency and Module Datapath Latency.
General	Added support for Microsoft CBL-Mariner Linux Operating System.
	Added a new flag, `[--skip_driver]`, to allow control of stop/start driver functionality in reset flow.
fwtrace	Added fwtrace the ability to detect when a token is appliec work in relevant mode (instead of constantly working in Se Mode).

MFT Bug Fixes History

The table below lists the history of bugs fixed. For a list of old Bug Fixes, please see MFT Archived Bug Fixes file.

Internal Ref. No.	Issue
4406754	**Description:** Fixed an issue with wrong device enumeration.
	Keywords: Enumeration
	Discovered in Version: 4.32.0-120
	Fixed in Release: 4.33.0-169
4398750	**Description:** Added support for reset level 1 type 3 and 4 when the driver is signed in a UEFI system.
	Keywords: UEFI

Internal Ref. No.	Issue
	Discovered in Version: 4.32.0-120
	Fixed in Release: 4.33.0-169
4378652	**Description:** Added support for SIGHUP signal handling in MFT infrastructure to trigger configuration reloads or graceful runtime updates.
	Keywords: SIGHUP
	Discovered in Version: 4.26.0
	Fixed in Release: 4.32.0-120
4337526	**Description:** Fixed an issue relating to querying the counters for local_port 255 in Spectrum4 standalone mode.
	Keywords: local_port 255, Spectrum4
	Discovered in Version: 4.29.0-131
	Fixed in Release: 4.32.0-120
4294588	**Description:** Changed the polling method to check if BlueField reset is complete.
	Keywords: BlueField
	Discovered in Version: 4.31.0-149
	Fixed in Release: 4.32.0-120
4291685	**Description:** Removed PSID MT_0000000884 from PCIE switch map.
	Keywords: PSID, PCIE
	Discovered in Version: 4.31.0-149
	Fixed in Release: 4.32.0-120
4290640	**Description:** Fixed the issue of handling bad arguments in mlxreg. The tool will report the error, print the usage message and exit.
	Keywords: mlxreg, bad error code
	Discovered in Version: 4.31.0-149
	Fixed in Release: 4.32.0-120
4258362	**Description:** Adapted the NIC driver name to support any custom name.
	Keywords: NIC driver name

Internal Ref. No.	Issue
	Discovered in Version: 4.31.0-149
	Fixed in Release: 4.32.0-120
4220920	**Description:** MFT on FreeBSD OS supports PCI-CORE VSC address spaces separation.
	Keywords: FreeBSD OS, PCI-CORE VSC address
	Discovered in Version: 4.30.0-139
	Fixed in Release: 4.31.0-149
4091646	**Description:** Changed the field name in code from dp_st_lane0 to dp_st_lane[0].
	Keywords: mlxlink,, DataPath,dp_st_lane0, dp_st_lane[0]
	Discovered in Version: 4.28.0
	Fixed in Release: 4.30.0
4078946	**Description:** Added handling interrupts between burning chunks to cables.
	Keywords: interrupts, burning chunks
	Discovered in Version: 4.28.0
	Fixed in Release: 4.30.0
3953342	**Description:** Removed support for OpenSSL engine.
	Keywords: OpenSSL engine
	Discovered in Version: 4.29.0
	Fixed in Release: 4.30.0
3832522	**Description:** Created VMware domain security policy according to the needs of MFT plugins.
	Keywords: VMware
	Discovered in Version: 4.26.1.102
	Fixed in Release: 4.29.0
3993256	**Description:** Corrected the order of reading data via I2C from big endian to little endian
	Keywords: I2C
	Discovered in Version: 4.28.0

Internal Ref. No.	Issue
	Fixed in Release: 4.29.0
3880918	**Description:** Added a validation that states that the provided path is a file and provided a better error message.
	Keywords: Validation, path
	Discovered in Version: 4.28.0
	Fixed in Release: 4.29.0
3953339	**Description:** Added a warning message before deprecating the support for OpenSSL engine in the next release.
	Keywords: OpenSSL
	Discovered in Version: 4.28.0
	Fixed in Release: 4.29.0
3886315	**Description:** To reset or shut down the BlueField Arm, it is mandatory to specify the --sync 0 argument with reset level 1 and reset type 3 or 4. For example: 'mlxfwreset -d -l 1 -t 4 --sync 0 r'.
	Keywords: BlueField Arm, shutdown
	Discovered in Version: 4.28.0
	Fixed in Release: 4.29.0
3819004	**Description:** flint's CMIS burn flow for cables ignores signals (e.g. CTRL+C) during the burn.
	Keywords: CMIS, burn
	Discovered in Version: 4.28.0
	Fixed in Release: 4.29.0
3645548	**Description:** Fixed an issue that led to wrong M1-M4 measurements calculation when using ConnecX-7.
	Keywords: M1-M4 measurements, ConnecX-7
	Discovered in Version: 4.26.0
	Fixed in Release: 4.26.1
3632765	**Description:** Fixed an issue that occurred when attempting to unzip the mlxfwmanager self-extractor.
	Keywords: mlxfwmanager, unzipping

Internal Ref. No.	Issue
	Discovered in Version: 4.25.0
	Fixed in Release: 4.26.0
3582574	**Description:** Fixed an issue that prevented ConnectX-5 EX reset using the fastfwreset tool.
	Keywords: fastfwreset
	Discovered in Version: 4.25.0
	Fixed in Release: 4.26.0
3582575	**Description:** Fixed an issue that caused incorrect enumeration in NVIDIA devices.
	Keywords: incorrect enumeration
	Discovered in Version: 4.25.0
	Fixed in Release: 4.26.0
3613010	**Description:** Fixed an issue where mlxdump did not work with Quantum-2 switches due to the absence of the Quantum2.csv file from the C:\Program Files\Mellanox\WinMFT\mstdump_dbs\ folder.
	Keywords: mlxdump, Quantum-2
	Discovered in Version: 4.25.0
	Fixed in Release: 4.26.0
3474570	**Description:** Fixed an issue that occurred when two MFT tools were running simultaneously, and while one of them loaded the driver, the second tool, which was not supposed to reload it in this case, failed to discover the Mellanox devices.
	Keywords: Driver reload
	Discovered in Version: 4.24.0
	Fixed in Release: 4.25.0
3471307	**Description:** Fixed an issue where incorrect eye information was displayed for 10G speed over ConnectX-7 devices.
	Keywords: Eye information, ConnectX-7
	Discovered in Version: 4.23.0
	Fixed in Release: 4.25.0

User Manual Revision History

Revision	Date	
4.34.0-145	November 10, 2025	Updated: • Access to Hardware Devices • Supported Operating Systems and Platforms • mlxconfig – Changing Device Configuration Tool - Added --with_default, --host_id and --pf_index flags • mlxlink Utility - Added --rx_modulation and --tx_modulation flags • mlxreg Utility - Added recommendation • mlxdpa – DPA Applications Sign Tool • Examples of mlxconfig Usage • Burning a Firmware Image • Verifying the Firmware Image • Querying the Firmware Image
4.33.0-169	August 08, 2025	Updated: • mlxlink Utility • mlxconfig – Changing Device Configuration Tool - Added --i2c_secondary • mlxreg Utility - Added --full_path • mlxfwreset – Loading Firmware on 5th Generation Devices Tool - Added PCIe Switch Reset via Hot Reset • flint – Firmware Burning Tool - Added component_type and skip_if_same • mlxfwmanager – Firmware Update and Query Tool - Added --component_type componentName and --skip_if_same
4.32.0-120	May 08, 2025	Updated: • Secure Host • mlxptrace Utility - Added --raw_dump flag • fwtrace Utility - Added --raw_dump flag • Burning a Firmware Image - Added Burning PLDM Images • Updating the Device using PLDM • mlxburn – Firmware Image Generator and Burner • mlxtokengenerator - Token Creation Tool • mlxburn – Firmware Image Generator and Burner

Revision	Date	
		• Linux - Added i2c • mlxfwmanager – Firmware Update and Query Tool • mlxcables - Cables Tool - Added note
4.31.0-149	February 10, 2025	Updated: • mget_temp Utility • mlxconfig – Changing Device Configuration Tool • Linux • Using mlxconfig • Secure Host Added: • ESXi
4.30.0-139	November 11, 2024	Updated: • mlxreg Utility • resourceparse Utility • Supported Operating Systems and Platforms • mlxconfig – Changing Device Configuration Tool Added: • mlxptrace Utility
4.29.0	August 14, 2024	Updated: • flint – Firmware Burning Tool • cpldupdate – Tool for Programming On-Board CPLDs on NVIDIA Devices • mlxlink Utility
4.28.0	May 7, 2024	Updated: • mget_temp Utility
4.27.0	February 8, 2023	Added Bifurcation Configuration under mlxconfig – Changing Device Configuration Tool.
4.26.1	December 14, 2023	No changes to the User Manual.

Revision	Date	
4.26.0	November 6, 2023	Updated: • mlxdpa – DPA Applications Sign Tool • mlxconfig – Changing Device Configuration Tool
4.25.1	October 22, 2023	Added: • Advance Options to "Generating an XML Template for the Configurations" under mlxconfig Commands.
4.25.0	August 10, 2023	Updated: • Cable Burn Command Running CMIS Firmware Upgrade Flow for Supported Cables under Burning a Firmware Image. • mlxdpa – DPA Applications Sign Tool • mlxfwreset – Loading Firmware on 5th Generation Devices Tool • fwtrace Utility • mlxtrace Utility • resourcedump Utility • resourceparse Utility • mst Synopsis - Windows under Windows. • mst Synopsis - VMware under VMware ESXi. • mst Synopsis - FreeBSD under FreeBSD. • mst Synopsis - Linux under Linux. Added: • mlxcableimgen - Cable Firmware Image Wrapper Generation Tool under Firmware Generation, Configuration, and Update Tools.

Document Conventions and Related Documents

Abbreviations and Acronyms

Term	Description
MFT	NVIDIA® Firmware tools
MST	Software tools and it is the name of the script that starts/stops the driver used by MFT tools
mlx	Extension of the text firmware file which contains all the firmware content
ini	Extension of the firmware configuration file which is in INI format and contains card specific configurations.
bin	Extension of the binary firmware file which is a combination of INI and mlx file
MFA	Extension of the a firmware file that contains several binary files of firmware for different cards/boards
4th Generation ICs/ Group I of ICs	Contains the following devices: • ConnectX-3 • ConnectX-3 Pro
5th Generation ICs/ Group II of ICs	Contains the following devices and newer: • Connect-IB • Switch-IB • Switch-IB 2 • NVIDIA Spectrum • NVIDIA Spectrum-2 • ConnectX-4 • ConnectX-4 Lx • ConnectX-5 • ConnectX-5 Ex • ConnectX-6

Term	Description
	• ConnectX-6 Dx • NVIDIA BlueField

Reference Documents and Downloads

Reference Documents and Downloads	Location
MFT web page	https://network.nvidia.com/products/adapter-software/firmware-tools/
Firmware Release Notes	https://docs.nvidia.com/networking/category/adapterfw
DOCA-Host	https://docs.nvidia.com/networking/dpu-doca/index.html#doca
WinOF/WinOF-2	https://docs.nvidia.com/networking/category/winof2
VMware	https://docs.nvidia.com/networking/category/vmwareesxiasyncdrivers
FreeBSD	https://docs.nvidia.com/networking/category/freebsd

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
Firmware	xx.47.1012	• HCA Firmware EULA • 3rd Party Unify Notice • 3rd Party Notice
DOCA-Host	3.2.0	• License • 3rd Party Unify Notice
MFT FreeBSD	4.34.0-145	• 3rd Party Notice • License
MFT Linux		• 3rd Party Notice • License
MFT VMware		• 3rd Party Notice • License
MFT Windows		• 3rd Party Notice • License