



NVIDIA Firmware Tools (MFT)

Documentation v4.18.0

Table of Contents

Overview	8
Intended Audience	8
Software Download	8
Document Revision History	8
Release Notes.....	9
Release Notes Update History.....	9
General Information	9
Package Tools	9
Software Dependencies	12
Dependencies.....	13
Supported Operating Systems and Platforms	14
Supported Flash Types.....	19
Supported Interface Cards (NICs).....	19
Supported Switch Systems Software	20
Changes and New Features.....	20
MFT Bug Fixes in this Version	22
MFT Known Issues	22
Introduction.....	28
Supported Operating Systems.....	28
Access to Hardware Devices.....	28
Compilation and Installation	31
Firmware Generation, Configuration, and Update Tools.....	32
mst Service	32
Linux	32
Running mst in an Environment without a Kernel.....	35
Windows.....	36
FreeBSD.....	37
VMware ESXi	38
MFT Configuration	40
MKey Configuration	40
mlxfwmanager - Firmware Update and Query Tool.....	40
mlxfwmanager Synopsis.....	40

Querying the Device	41
Archived Images Content	42
Updating the Device	43
UPMF	45
UPMF Generation Flow	45
Updating Firmware Using an UPMF	48
mlxarchive - Binary Files Compression Tool.....	48
mlxarchive Synopsis	49
mlxconfig - Changing Device Configuration Tool	49
Tool Requirements	49
mlxconfig Synopsis.....	50
Examples of mlxconfig Usage	51
Using mlxconfig	53
mlxconfig Commands.....	57
MFT Supported Configurations and Parameters	59
flint - Firmware Burning Tool	60
flint Synopsis	60
Switches Options.....	60
Command Parameters.....	64
Burning a Firmware Image.....	66
Querying the Firmware Image	72
Verifying the Firmware Image	73
Performing Checksum Calculation on Image/Device	73
Managing an Expansion ROM Image	74
Setting GUIDs and MACs.....	75
Setting the VSD.....	79
Disabling/Enabling Access to the Hardware	80
Flash Operations	81
Firmware Timestamping for Multi-Host Environment	83
flint/mlxburn Limitations.....	84
Secure Host	85
Secure Firmware Update	87
Secure Boot	92
mlxburn - Firmware Image Generator and Burner	93

Generating and Burning Firmware	93
Customizing Firmware	94
mlxburn Synopsis.....	94
Examples of mlxburn Usage	97
Exit Return Values	100
mlxfwreset - Loading Firmware on 5th Generation Devices Tool.....	100
Tool Requirements	100
Query Command	100
Reset Command	100
mlxfwreset Synopsis	100
Reset Levels and Types.....	101
Reset Sync.....	101
mlxfwreset for Multi-Host NICs	101
mlxfwreset for SmartNICs	102
Examples of mlxfwreset Usage	102
mlxfwreset Limitations	103
mlxphyburn - Burning Tool for Externally Managed PHY	103
Tool Requirements	103
mlxphyburn Synopsis	103
Examples of mlxphyburn Usage.....	104
mlx_fpga - Burning and Debugging Tool for NVIDIA Devices with FPGA.....	104
Tool Requirements	104
mlx_fpga Synopsis	104
Examples of mlx_fpga Usage	105
cpldupdate - Tool for Programming On-Board CPLDs on NVIDIA Devices	107
Tool Requirements	108
cpldupdate Synopsis	108
Burn Example.....	108
mstcongestion - Tool for Setting Congestion Mode and Action.....	108
Tool Requirements	108
mstcongestion Synopsis	109
mlxprivhost - NIC Configuration by the Host Restriction Tool.....	109
mlxprivhost Synopsis	109
Debug Utilities.....	111

fwtrace Utility.....	111
fwtrace Usage	111
itrace Utility.....	113
itrace Usage	113
mstdump Utility.....	115
mstdump Usage	115
mlx2c Utility	115
mlx2c Usage.....	115
i2c Utility.....	116
i2c Usage (Advanced Users)	116
Exit Return Values	117
mget_temp Utility	117
mget_temp Usage.....	117
mlxtrace Utility	118
mlxtrace Usage.....	118
mlxdump Utility	120
mlxdump Usage	120
mlxmcg Utility.....	121
mlxmcg Usage	121
pckt_drop Utility.....	122
pckt_drop Usage	122
mlxuptime Utility	123
mlxuptime Usage.....	123
wqdump Utility.....	123
wqdump Usage	124
mlxmdio Utility.....	126
mlxmdio Usage	126
mlxreg Utility.....	128
mlxreg Usage	128
mlxlink Utility	130
mlxlink Usage.....	130
Margin Scan Tool	135
RX Error Injection.....	136
Tool Usage with NIC vs. Switch (-p Flag)	136

Tool Usage on NVIDIA Quantum HDR Switch Systems with Split Ports	136
PCIe	137
resourcedump Utility	140
resourcedump Usage	140
resourceparse Utility	142
resourceparse Usage	142
stedump Utility.....	145
Prerequisites.....	145
stedump Usage	145
Cable Utilities	149
Cable Discovery	149
How to Discover the Cables	149
Representing the Cables in mst Status	149
Working with Cables.....	150
mlx cables - Cables Tool	150
mlx cables Synopsis	151
MTUSB Cable Board.....	154
Troubleshooting	155
General Related Issues	155
mlxconfig Related Issues.....	155
Installation Related Issues	156
Firmware Burning Related Issues	156
Secure Firmware Related Issues	158
Appendixes.....	160
Assigning PSID	160
PSID Field Structure	160
Assigning PSID and Integrating Flow	160
Remote Access to NVIDIA® Devices	161
Burning a Switch In-band Device Using mlxburn	161
In-Band Access to Multiple IB Subnets	162
MTUSB-1 USB to I2C Adapter	163
Remote Access to Device by Sockets.....	165
Accessing Remote InfiniBand Device by Direct Route MADs	167
Booting HCA Device in Livefish Mode	167

Booting Card in Livefish Mode	168
Booting Card in Normal Mode	168
Common Locations of Flash Present Pins	168
Burning a New Device	169
Connect-IB Adapter Card	169
ConnectX®-4 onwards Adapter Cards Family	171
Spectrum® or Switch-IB® 2 Switch Systems	173
Switch-IB® Switch System	175
Generating Firmware Secure and NV LifeCycle Configurations Files	177
Create Forbidden Versions Binary File	177
Create Tokens for Secure Firmware and NV LifeCycle	177
Updating Firmware Using ethtool/devlink and .mfa2 File.....	177
Document Conventions and Related Documents	180
Abbreviations and Acronyms	180
Reference Documents and Downloads	180
User Manual Revision History.....	182
Release Notes Revision History	184
MFT Release Notes Change Log History.....	184
MFT Bug Fixes History	205

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
4.18.0	December 05, 2021	Initial release of this Release Notes version, This version introduces Changes and New Features and 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.

The MFT tools do not provide cluster wide functionality.

Category	Tool	Description	Operating System
MST Service	mst	<ul style="list-style-type: none">Lists the available mst devicesStart/stop the register access driver for Linux and VMware ESXi OSs.	All
Firmware Update and Configuration	mlxburn	This tool provides the following functions: <ul style="list-style-type: none">Generating a standard or customized firmware image for burning in .bin formatBurning an image to the Flash attached to an HCA or switch deviceQuerying the firmware version loaded on a deviceDisplaying 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

Category	Tool	Description	Operating System
	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 with read/write registers in the QDR memory of the FPGA.	Linux
	cplupdate	A tool for programing on board CPLDs for NVIDIA® devices for the OEM packages only.	Linux

Category	Tool	Description	Operating System
Debug and Diagnostic 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
	mlx_i2c	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
	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

Category	Tool	Description	Operating System
	mlx cables	Reads/writes NVIDIA® cable registers and queries the cables info	All
	mlxlink	Displays and configures port related data at the physical layer.	All
	mlxvdp	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
OFED / MLNX_OFED ^{1, 2}	1.5.0 or later
Perl	5.24 or later
Python ³	2.6 or later
lsusb ⁴	
rpmbuild	
xz ⁵	
Windows	
NVIDIA® WinOF VPI ⁶	3.0.0 or later

VMware ESXi	
Python	2.6 or later

Notes:

1. OFED can be downloaded from <http://www.openfabrics.org>. Note that installing OFED 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 OFED packages "ibutils" or "ibutils2" or "infiniband-diags" should be installed and available in the PATH. (For details on OFED installation, visit <http://www.mellanox.com> and under OFED.)
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 from www.mellanox.com > Products > Software > InfiniBand Drivers (LEARN MORE) > Windows SW Drivers.
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 in MFT v4.18.0

Supported Operating Systems and Platforms

OS	Platform	Status
AliOS 7.2	aarch64	+
BCLinux 7.5	x86_64	+
BCLinux 7.6	x86_64	+
BCLinux 8.1	x86_64	+
Debian 10	x86_64	+
Debian 10.3	x86_64	+
Debian 10.5	x86_64	+
Debian 10.9	x86_64	+
Debian 9.11	x86_64	+
Debian 9.9	x86_64	+
EulerOS V2.0 SP10	x86_64	**
EulerOS V2.0 SP5	x86_64	+
EulerOS V2.0 SP8	x86_64	**
EulerOS V2.0 SP9	x86_64	+
Fedora 32	X86_64	+
FreeBSD 12-STABLE (12.2)	x86_64	+
FreeBSD 13.0-STABLE	x86_64	+
FreeBSD 14-CURRENT	x86_64	+

OS	Platform	Status
KylinOS v10 SP2	x86_64	**
MLNX-OS	64 Bit	+
OEL 7.8	x86_64	+
OEL 7.9	x86_64	+
OEL 8.2	x86_64	+
OEL 8.3	x86_64	+
OpenEuler 20.3	x86_64	+
OpenEuler 20.3 SP1	x86_64	+
RHEL 7.2	x86_64	+
RHEL 7.3	x86_64	+
RHEL 7.4	x86_64	+
RHEL 7.4	PPC64	+
RHEL 7.4	PPC64LE	+
RHEL 7.4 Alt	ARM	**
RHEL 7.5	PPC64	+
RHEL 7.5	PPC64LE (Power 9)	+
RHEL 7.5	x86_64	+
RHEL 7.6	ARM	+
RHEL 7.6	PPC64	+
RHEL 7.6	x86_64	+
RHEL 7.7	PPC64LE	**
RHEL 7.7	x86_64	+
RHEL 7.7	PPC64	+
RHEL 7.8	x86_64	+

OS	Platform	Status
RHEL 7.8	PPC64	+
RHEL 7.8	PPC64LE	+
RHEL 7.9	x86_64	**
RHEL 7.9	PPC64	**
RHEL 7.9	PPC64LE	**
RHEL 8	ARM	**
RHEL 8	PPC64LE	**
RHEL 8	x86_64	+
RHEL 8.1	PPC64LE (Power 9)	**
RHEL 8.1	x86_64	+
RHEL 8.1	ARM	+
RHEL 8.2	x86_64	+
RHEL 8.2	PPC64LE	+
RHEL 8.3	PPC64LE	+
RHEL 8.3	x86_64	+
RHEL 8.4	x86_64	+
RHEL 8.4	PPC64LE	+
RHEL 8.5	PPC64LE	**
RHEL 8.5	x86_64	+
Sles12 SP2	PPC64LE	**
Sles12 SP2	x86_64	+
Sles12 SP3	x86_64	+
Sles12 SP3	ppc64LE	+
Sles12 SP4	ARM64	**

OS	Platform	Status
Sles12 SP4	PPC64LE	**
Sles12 SP4	x86_64	+
Sles12 SP5	ARM64	**
Sles12 SP5	x86_64	+
Sles12 SP5	PPC64LE	+
Sles15 SP1	PPC64LE	**
Sles15 SP1	ARM64	**
Sles15 SP1	x86_64	+
Sles15 SP2	x86_64	+
Sles15 SP2	PPC64LE	+
Sles15 SP3	PPC64LE	+
Sles15 SP3	x86_64	+
SONiC	64 Bit	+
Ubuntu 14.04.06	x86_64	+
Ubuntu 16.04	x86_64	+
Ubuntu 16.04	PPC64LE	+
Ubuntu 18.04	x86_64	+
Ubuntu 18.04	PPC64LE	+
Ubuntu 18.04	ARM64	+
Ubuntu 20.04	PPC64LE	+
Ubuntu 20.04	ARM64	+
Ubuntu 20.04	x86_64	+
Ubuntu 21.04	x86_64	+
Ubuntu 21.10	x86_64	+

OS	Platform	Status
UOS v20 1030	x86_64	+
VMware ESXi 6.5 Native	64 Bit	+
VMware ESXi 6.7u2 Native	64 Bit	+
VMware ESXi 7.0 Native (Vsphere 2019)	64 Bit	+
VMware ESXi 7.0 u2 Native (Vsphere 2019)	64 Bit	+
VMware ESXi 7.0 u3 Native (Vsphere 2019)	64 Bit	**
VMware ESXi 7.0 u3 Native (Vsphere 2019)	64 Bit	+
Windows 10 1809	64 Bit	+
Windows 10 2022	64 Bit	+
Windows 8.1	64 Bit	+
Windows AH Server AH21	64 Bit	+
Windows latest SAC (windows 10 Client 21H1)	64 Bit	+
Windows Server 2012	64 Bit	+
Windows Server 2012R2	64 Bit	+
Windows Server 2016	64 Bit	+
Windows Server 2019	ARM64	+
Windows Server 2019	64 Bit	+
Windows Server 2022	64 Bit	+
WinPE 10	32 Bit	**
WinPE 10	64 Bit	**
WinPE 10	32 Bit	+
WinPE 10	64 Bit	+
WinPE 4.0	32 Bit	+
WinPE 4.0	64 Bit	+

OS	Platform	Status
WinPE 5.0	32 Bit	+
WinPE 5.0	64 Bit	+
WinPE 5.1	32 Bit	+
WinPE 5.1	64 Bit	+

Supported Flash Types

MFT supports the following Flash types.

Vendor	Flash Family	Tested P/N
Winbond	W25QxxBV	W25Q32FVSSIG
		W25Q32FVSSIGS
		W25Q32FVSSIGT
		W25Q128FVSSIGS
	W25Qxxx	W25Q256JVBIMT
		W25Q128JVSIG
Macronix	MX25L16xxx	MX25L12845GM2I-08G
	MX25Lxxx	MX25L25645GXDI-08G
Micron	N25QOxxx	MT25QL128ABA1ESE-0SIT
ISSI	IS25LPxxx	IS25LP128-JBLE SPA# U1323A
	IS25WPxxx	IS25WP256E-RHLE
Cypress	S25FL256L	S25FL256LDPBHV023
Gigadevice		GD25LB256EBFRY

Supported Interface Cards (NICs)

With respect to MFT, NVIDIA NIC devices are divided into two groups: Group I and Group II (4th generation and 5th generation, respectively). The ICs are listed in the following table:

IC Group	IC Device
Group II / 5th Generation	<ul style="list-style-type: none"> • Adapter Cards: <ul style="list-style-type: none"> • NVIDIA® BlueField-2® • NVIDIA® BlueField® • NVIDIA® ConnectX®-6 Lx • NVIDIA® ConnectX®-6 Dx • NVIDIA® ConnectX®-6 • NVIDIA® ConnectX®-5 • NVIDIA® ConnectX®-4 Lx • NVIDIA® ConnectX®-4 • NVIDIA® Connect-IB® • Switch Systems: <ul style="list-style-type: none"> • NVIDIA Quantum • NVIDIA Spectrum™-2 • NVIDIA Spectrum™ • NVIDIA® Switch-IB® 2 • NVIDIA® Switch-IB®
Group I / 4th Generation	<ul style="list-style-type: none"> • Adapter Cards: <ul style="list-style-type: none"> • NVIDIA® ConnectX®-3 • NVIDIA® ConnectX®-3 Pro • Switch Systems: <ul style="list-style-type: none"> • NVIDIA® SwitchX®-2

Supported Switch Systems Software

The following are the Supported Switch Systems Software.

Switch Software	Version
MLNX-OS	3.9.2000 and above
SONIC	

Changes and New Features

Changes and New Features in this Release

Component/ Tool	Description	Operating System
Rev. 4.18.0		
Python 2.x	<p>Python 2.x is now end-of-life and no longer supported by MFT.</p> <p>To use the latest and up-to-date MFT tools, we recommend you use Python 3.x.</p>	All

Component/ Tool	Description	Operating System
Rev. 4.18.0		
flint	When downgrading to a firmware version that does not support the flash type of the device, flint will present the user a clear error of such scenario.	All
mlxfwreset	Added a new reset-type ("NIC only reset") to mlxfwreset which is applicable only to SmartNIC devices. The new reset-type is also the new default for SmartNIC devices. In case of reset-type is set to "NIC only reset", mlxfwreset will not reset the internal host.	All
mlxlink	Added support for mlxlink on Windows Arm64 architecture. For further information, see mlxlink Utility section.	Windows
mlxlink	Added support for new PRBS TX/RX patterns (--tx_prbs <tx_prbs_mode> & --rx_prbs <rx_prbs_mode>). For further information, see mlxlink Utility section.	All
mlxlink	Added new show counters for 16nm devices. To see them run the "show_counter" command.	All
mlxlink	Extended the list of the cable information received for 16nm devices when running the "show_module" command.	All
mlxlink	Extended the information collection for 7nm and 16nm devices. See "--amber_collect" flag. For further information, see mlxlink Utility section.	All
mlxlink	Extended the list of the cable information (LOL, LOS, FSM, and module status) for CMIS when running the "show_module" command.	All
mlxlink	Added support for InfiniBand operations in the mlxlink tool. Now HCA devices can be accessed via the InfiniBand protocol.	All
MADs	MFT tools will now use class 0xA instead of class 9 for ConfigSpaceAccess MADs.	All
Arm Support	Added support for arm64 architecture to the WinMFT package.	Windows
ESXiO	MFT is now supported on NVIDIA BlueField (Arm) in VMware ESXiO environments.	All
Vendor Specific Key Security	Added support for Vendor Specific Key Security. Vendor Specific Keys are an authentication mechanism for using GMP MADs.	All

Component/ Tool	Description	Operating System
Rev. 4.18.0		
Bug Fixes	See Bug Fixes .	All

MFT Bug Fixes in this Version

This version does not include Bug Fixes. For a list of old Bug Fixes, please see [MFT Bug Fixes History](#).

Internal Ref.	Description
2850979	Description: Fixed mlxlink PCIe link validation flow on NVIDIA BlueField controller mode.
	Keywords: mlxlink, PCIe, NVIDIA BlueField
	Discovered in Version: 4.17.0
	Fixed in Release: 4.18.0
2578359	Description: Using Phyless reset with level 4 (warm reboot with NIC phyless reset) may result in hardware errors and link dropping.
	Keywords: Phyless reset
	Discovered in Version: 4.17.0
	Fixed in Release: 4.18.0
2665520	Description: Issuing mlxlfwreset -l 4 in a multihost system or within a DPU device will lead to the host reboot without affecting the device.
	Keywords: Multihost; host reboot
	Discovered in Version: 4.17.0
	Fixed in Release: 4.18.0

MFT Known Issues

The following table provides a list of known issues and limitations. For a list of old Known Issues, please see [MFT Archived Known Issues](#) file.

Internal Ref. No.	Issue
2871042	Description: mlxfwmanager default query on switches will take pci_cr0 instead of pciconf0, which is expected to fail in secure-fw switches.
	Workaround: Specify the switch's mst name to the mlxfwmanager by giving it the pciconf0 value.
	Keywords: mlxfwmanager, pci_cr0, pciconf
	Discovered in Version: 4.18.0
2787479	Description: mlx cables shows the wrong firmware version for OSFP cables.
	Workaround: N/A
	Keywords: mlx cables, 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 devices.
	Workaround: N/A
	Keywords: mlxfwreset
	Discovered in Version: 4.18.0
2752916	Description: The information of the IB/ETH protocols should not be stored on the same CSV file. Doing so will result in a mismatch on the columns of CSV file.
	Workaround: N/A
	Keywords: mlxlink
	Discovered in Version: 4.18.0
2838222	Description: mlxfwreset is not supported on kernel 3.10.0-1062.el7.x86_64 due to a kernel bug that leads to 'rescan' PCI operation to take a few minutes.
	Workaround: N/A
	Keywords: mlxfwreset
	Discovered in Version: 4.18.0

Internal Ref. No.	Issue
2703663	Description: Running flint commands on the hypervisor while a Virtual Machine is running with the same device (pass-through), may cause kernel panic.
	Workaround: N/A
	Keywords: flint, kernel, VM
	Discovered in Version: 4.17.0
2670833	Description: Burning firmware using DMA might fail on virtual FreeBSD machines.
	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 lanes as shown below: <ul style="list-style-type: none"> • 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
2580945	Description: Host reboot may reboot the Arm side as well if the device's configuration is done via mlxconfig.
	Workaround: N/A
	Keywords: Non-Volatile configuration, Arm, reboot
	Discovered in Version: 4.16.3
2392334	Description: Using the MFT with the --with-pcap option to install stedump utility requires the following third-party dependencies: <ul style="list-style-type: none"> • 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 following: <ol style="list-style-type: none"> 1. Install libpcap-devel or libpcap-dev on Debian-based distributions. 2. Install python3-devel or python3-dev on Debian-based distributions. 3. Bootstrap the PIP installer in one of the following ways: <ul style="list-style-type: none"> • On Python 3.4 or newer bootstrap it from the standard library by running the "python -m ensurepip" command. • Install PIP with Linux Package Manager, for more details see: https://packaging.python.org/guides/installing-using-linux-tools/
	Keywords: stedump utility

Internal Ref. No.	Issue
	Discovered in Version: 4.16.0
2376425	Description: Direct Device Assignment (DDA, ak.a. pass-through) facility is not supported in MFT, its usage may cause the host to reboot.
	Workaround: Burn the firmware in PF and then attach the HCA to the VM.
	Keywords: DDA
	Discovered in Version: 4.16.0
2208845/2099263	Description: mlxlink does not support test mode for 50GE-KR4 speed.
	Workaround: N/A
	Keywords: mlxlink
	Discovered in Version: 4.16.0
2274123	Description: mlxfwreset is supported on SmartNic devices on Windows OS only if the device's name format is "mt*_pciconf*" and not "**:*.*".
	Workaround: N/A
	Keywords: mlxfwreset
	Discovered in Version: 4.16.0
-	Description: Port toggling with Inband devices using mlxlink fails and the following error is presented: "Unknown MAD error".
	Workaround: To avoid this issue, perform one of the following options: <ul style="list-style-type: none"> • 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 causes network connectivity disconnection.
	Workaround: Maintain another interface for enabling the link back.
	Keywords: mlxlink
	Discovered in Version: 4.15.0

Internal Ref. No.	Issue
2167841	Description: "mlxfwmanager --download" and "mlxfwmanager --online" commands are currently not functional on ESXi 7.0.
	Workaround: N/A
	Keywords: mlxup/mlxfwmanager
	Discovered in Version: 4.14.3
2149437	Description: When the SLTP configuration is wrongly set, the “Bad status” explanation will not be presented (only error indication) to the user.
	Workaround: N/A
	Keywords: SLTP configuration
	Discovered in Version: 4.14.2
1780276	Description: "mst server start" runs at foreground instead of the background on FreeBSD and VMWare ESXi OSes.
	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 versions =>2.7 and >=3.2. In case an older Python version is used, the argparse module is not installed by default and therefore must be manually 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 ConnectX-6 InfiniBand adapter cards.
	Workaround: N/A
	Keywords: mlxlink, Force Mode, ConnectX-6 IB
	Discovered in Version: 4.13.3
1923665	Description: mlxburn is not signed for Windows operating systems.
	Workaround: N/A
	Keywords: mlxburn, Windows, operating system, signature
	Discovered in Version: 4.13.0
1802662	Description: Due to mst signing process, some executions might be slower than expected.
	Workaround: N/A
	Keywords: mst
	Discovered in Version: 4.13.0

Internal Ref. No.	Issue
1431471	Description: In ConnectX-5 adapter cards, the time-stamp capability using flint, is supported only on the device using the "-d" flag, and not on the binary using the "-i" flag.
	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-Host device, the driver remains down if the mstfwreset operation fails.
	Workaround: N/A
	Keywords: mstfwreset
	Discovered in Version: 4.11.0
-	Description: Running mstfwreset on ConnectX-5 Socket-Direct adapter cards on Windows OS is currently not functional.
	Workaround: Reboot the server
	Keywords: mstfwreset, ConnectX-5 Socket-Direct
	Discovered in Version: 4.8.0

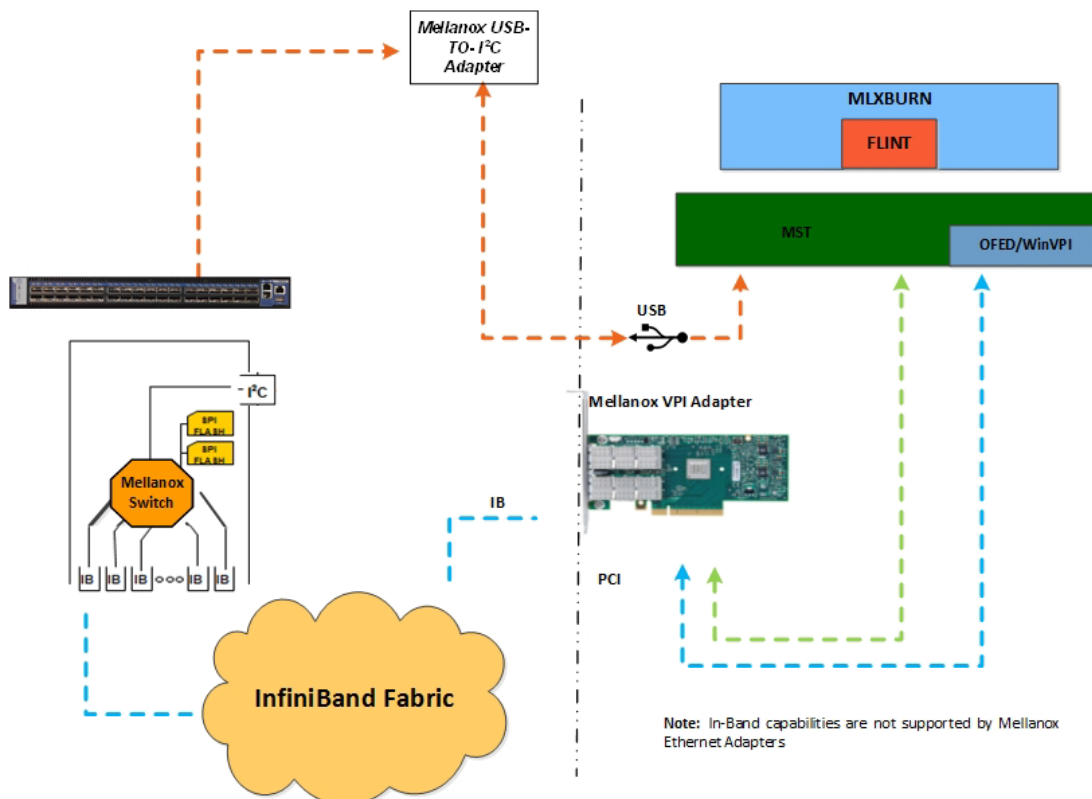
Introduction

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 Mellanox 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 platforms and kernels.


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

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
HCA (InfiniBand)	NVIDIA® Connect-IB	V	V	V
VPI Network Adapter	NVIDIA® ConnectX®-3	V	V	V
	NVIDIA® ConnectX®-3 Pro	V	V	V
	NVIDIA® ConnectX®-4	V	V	V
	NVIDIA® ConnectX®-5	V	V	V
	NVIDIA® ConnectX®-5 Ex	V	V	V
	NVIDIA® ConnectX®-6	V	V	V
Ethernet Adapter (NIC)	NVIDIA® ConnectX®-3 EN	V	V	
	NVIDIA® ConnectX®-4 Lx	V	V	
	NVIDIA® ConnectX®-6 Dx	V	V	
	NVIDIA® ConnectX®-6 Lx	V	V	
Switch	NVIDIA® SwitchX®-2	V ²	V	V
	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

Note. V¹ 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).

 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](#) 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)
```

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	<ol style="list-style-type: none"> 1. Untar the downloaded package 2. Allow packaging of bins to a self-executing file. 3. Run 'install.sh' 4. For OEM only: 'install.sh --oem' 4. Start the mst driver by running: mst start <p>NOTE: It is possible to customize some installation parameters (such as the target installation path). Run 'install.sh --help' for details.</p>	Uninstall MFT on Linux by running the following command: mft_uninstall.sh
Windows	<p>The installation is EXE based:</p> <ol style="list-style-type: none"> 1. Double click the EXE file and follow the instructions presented by the installation wizard. 	<ol style="list-style-type: none"> 1. Go to Add or remove programs. 2. Remove WinMFT64 depending on the platform type.
FreeBSD	<ol style="list-style-type: none"> 1. Untar the downloaded package. 2. Run "install.sh" 	Uninstall MFT on FreeBSD, run the following command: mft_uninstall.sh
VMware	<ol style="list-style-type: none"> 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 	<ol style="list-style-type: none"> 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](#)
- [Secure Boot](#)
- [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](#)
- [mstcongestion - Tool for Setting Congestion Mode and Action](#)
- [mlxprivhost - NIC Configuration by the Host Restriction Tool](#)

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.

	<ul style="list-style-type: none"> • <code>--with_i2cdev</code>: Create Embedded I2C master • <code>--with_fpga</code>: Create MST device for the attached FPGA card (Access via Driver) • <code>--with_fpga_fw_access</code>: Create an extended MST device for the attached FPGA (Access via Firmware)
<code>mst stop [--force]</code>	<p>Stop the MST driver service, remove all special files/ directories and unload kernel modules.</p> <p>Options:</p> <ul style="list-style-type: none"> • <code>--force</code>: Force try to stop mst driver even if it's in use.
<code>mst restart [--with_msix] [--with_unknown] [--with_i2cdev] [--with_lpcdev] [--with_fpga] [--with_fpga_fw_access]</code>	<p>Just like "mst stop" followed by "mst start [--with_msix] [--with_unknown] [--with_i2cdev] [--with_lpcdev] [--with_fpga] [--with_fpga_fw_access]"</p>
<code>mst server start [port]</code>	<p>Start mst server to allow incoming connection. Default port is 23108</p>
<code>mst server stop</code>	<p>Stop mst server.</p>
<code>mst remote add <hostname>[:port]</code>	<p>Establish connection with specified host on specified port (default port is 23108). Add devices on remote peer to local devices list. <hostname> may be host name as well as an IP address.</p>
<code>mst remote del <hostname>[:port]</code>	<p>Remove all remote devices on specified hostname.</p> <p><hostname>[:port] should be specified exactly as in the "mst remote add" command.</p>
<code>mst ib add [OPTIONS] [local_hca_id] [local_hca_port]</code>	<p>Add devices found in the IB fabric for inband access. Requires OFED installation and an active IB link.</p> <p>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.</p> <p>Options:</p> <ul style="list-style-type: none"> • <code>--discover-tool <discover-tool></code>: The tool that is used to discover the fabric. <p>Supported tools: ibnetdiscover, ibdiagnet. default: ibdiagnet</p> <ul style="list-style-type: none"> • <code>--add-non-mlnx</code>: Add non NVIDIA® nodes. • <code>--topo-file <topology-file></code>: A prepared topology file which describes the fabric. For ibnetdiscover: provide an output of the tool. For ibdiagnet: provide LST file that ibdiagnet generates. • <code>--use-ibdr</code>: Access by direct route MADs. Available only when using ibnetdiscover tool, for SwitchX and ConnectIB devices. <p>NOTE: If a topology file is specified, device are taken from it. Otherwise, a discover tool is run to discover the fabric.</p>
<code>mst ib del</code>	<p>Remove all inband devices.</p>

mst cable add [OPTIONS] [params]	<p>Add the cables that are connected to 5th generation devices.</p> <p>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.</p> <p>Options:</p> <p>--with_ib: Add the inband cables in addition to the local PCI devices.</p> <p>params: [local_hca_id] [local_hca_port]</p>
mst cable del	Remove all cable devices.
mst status	<p>Print current status of NVIDIA® devices</p> <p>Options:</p> <ul style="list-style-type: none"> -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	<ul style="list-style-type: none"> 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:

```
Success# 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
```

 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
Base GUID:       7cfe90030029205e  GuidsNumber 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

 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]	Establish connection with specified host on specified port (default port is 23108). Add devices on remote peer to local devices list. <hostname> may be host name as well as an IP address.
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 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.

<pre>mst ib add [OPTIONS] [local_hca_id] [local_hca_port] [lst-file]</pre>	<p>Add devices found in the IB fabric for inband access. 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.</p> <p>Options:</p> <ul style="list-style-type: none"> • --discover-tool <discover-tool>: The tool used to discover the fabric. <p>Supported tools: ibnetdiscover, ibdiagnet. default: ibnetdiscover</p> <p>NOTE: The discover tool argument is intended only for parsing purposes, thus you need to specify an lst-file with it.</p> <ul style="list-style-type: none"> • --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. <p>NOTE: If a topology file is specified, the devices are taken from it. Otherwise, a discover tool is run to discover the fabric.</p>
<pre>mst help</pre>	<p>Print this help information.</p>

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:

```
FreeBSD# 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


 There are no mst start or stop operations in FreeBSD.

mst status	Print current status of Mellanox 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 Mellanox devices:

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

 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 port]	Start mst server to allow incoming connection. Default port is 23108.
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>`.

 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 [Bootting HCA Device in Livefish Mode](#).

 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-downloadopt OPT] [--download-device Device] [--download-os OS] [--download-type Type] [--ssl-certificate
Certificate] [--no_fw_ctrl]
```

where:

<code>-d --dev DeviceName</code>	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.
<code>-h --help</code>	Show this message and exit.
<code>-v --version</code>	Show the executable version and exit.
<code>--query</code>	Query device(s) info
<code>--query-format Format</code>	(Query Onlinequery)outputformat,XML Text-defaultText
<code>-u --update</code>	Update firmware image(s) on the device(s).
<code>-i --image-file FileName</code>	Specified image file to use.
<code>-D --image-dir DirectoryName</code>	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

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 09:00.0 --query
Querying Mellanox devices firmware ...

Device #1:
-----
Device Type:      ConnectX3
Part Number:      MCX354A-FCA_A2-A4
Description:      ConnectX-3 VPI adapter card; dual-port QSFP; FDR IB (56Gb/s) and 40GigE; PCIe3.0 x8 8GT/s; RoHS R6
PSID:             MT_1020120019
PCI Device Name:  0000:09:00.0
Port1 GUID:       0002c9000100d051
Port2 MAC:        0002c9000002
Versions:         Current      Available
FW               2.31.5050     2.32.5000
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:      ConnectIB
Part Number:      MCB192A-FCA_A1
Description:      Connect-IB Host Channel Adapter; single-port QSFP; FDR 56Gb/s; PCIe2.0 x16; RoHS R6
PSID:             MT_1220110030
PCI Device Name:  /dev/mst/mt4113_pciconf0
Port1 GUID:       0002c903002ef500
Port2 GUID:       0002c903002ef501
Versions:         Current      Available
FW               2.11.1258     10.10.4000
Status: Update required
-----

Device #2:
-----
Device Type:      ConnectX3
Part Number:      MCX354A-FCA_A2-A4
Description:      ConnectX-3 VPI adapter card; dual-port QSFP; FDR IB (56Gb/s) and 40GigE; PCIe3.0 x8 8GT/s; RoHS R6
PSID:             MT_1020120019
PCI Device Name:  /dev/mst/mt4099_pci_cr0
Port1 GUID:       0002c9000100d051
Port2 MAC:        0002c9000002
Versions:         Current      Available
FW               2.31.5050     2.32.5000
Status: Update required
-----
Found 2 device(s) requiring firmware update. Please use -u flag to perform the update.
```

Query XML:

```
mlxfwmanager --query --query-format XML
<Devices>
  <Device pciName="/dev/mst/mt4099_pci_cr0" type="ConnectX3" psid="MT_1200111023" partNumber="MCX354A-FCA_A2-A4">
    <Versions>
      <FW current="2.1.0065" available="2.32.5000"/>
    </Versions>
    <MACs port1="02c90abcdef0" port2="02c90abcdef1"/>
    <Status> update required </Status>
    <Description> ConnectX-3 VPI adapter card; dual-port QSFP; FDR IB (56Gb/s) and 40GigE; PCIe3.0 x8 8GT/s; RoHS R6
  </Description>
  </Device>
  <Device pciName="/dev/mst/mt4113_pciconf0" type="ConnectIB" psid="MT_1220110030" partNumber="MCB192A-FCA_A1">
    <Versions>
      <FW current="2.11.1258" available="10.10.4000"/>
    </Versions>
    <GUIDs port1="0002c903002ef500" />
    <MACs port1="0002c903002ef501" />
    <Status> update required </Status>
    <Description> Connect-IB Host Channel Adapter; single-port QSFP; FDR 56Gb/s; PCIe2.0 x16; 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>
```

Example:

```
mlxfwmanager -u -d 0000:09:00.0 -i fw-ConnectX3-rel-2.32.5000.mfa
Querying Mellanox devices firmware ...
Device #1:
-----
Device Type:      ConnectX3
Part Number:      MCX354A-FCA_A2-A4
Description:      ConnectX-3 VPI adapter card; dual-port QSFP; FDR IB (56Gb/s) and 40GigE; PCIe3.0 x8 8GT/s; RoHS R6
PSID:             MT_1020120019
PCI Device Name:  0000:09:00.0
Port1 GUID:       0002c9000100d051
Port2 MAC:        0002c90000002
Versions:         Current    Available
FW                2.31.5050   2.32.5000
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 0000:09:00.0 -y
Querying Mellanox devices firmware ...
Device #1:
-----
Device Type:      ConnectX3Pro
Part Number:      MCX354A-FCC_Ax
Description:      ConnectX-3 Pro VPI adapter card; dual-port QSFP; FDR IB (56Gb/s) and 40GigE; PCIe3.0 x8 8GT/s; RoHS R6
PSID:             MT_1090111019
PCI Device Name:  0000:09:00.0
Port1 GUID:       0002c90300e955e1
Port2 GUID:       0002c90300e955e2
Versions:         Current    Available
FW                2.32.5506   2.33.5000
PXE 3.4.0460 3.4.0460
Status: Update required
-----
Found 1 device(s) requiring firmware update...
Device #1 Release notes:
-----
Version 2.33.5000 contains the following features/bug fixes:
1- Virtual QoS support.
2- RX buffer optimizations for DSCP mode.
3- SMBUS ARP support.
4- RDMA Retransmission optimization.
5- NVCONFIG: UAR BAR change support.
6- Sideband connectivity improvements (IPMI, NCSI).
For full list of features and bug fixes please see full release notes at:
ConnectX3: http://www.mellanox.com/pdf/firmware/ConnectX3-FW-2\_33\_5000-release\_notes.pdf
ConnectX3Pro: http://www.mellanox.com/pdf/firmware/ConnectX3Pro-FW-2\_33\_5000-release\_notes.pdf
-----
Please wait while downloading MFA(s) 100%
```

```
Device #1: Updating FW ... 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/ConnectX4Firmwarev12282008
5- ConnectX4Lx (14.30.1004): https://docs.mellanox.com/display/ConnectX4LxFirmwarev14301004
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 Mellanox 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.

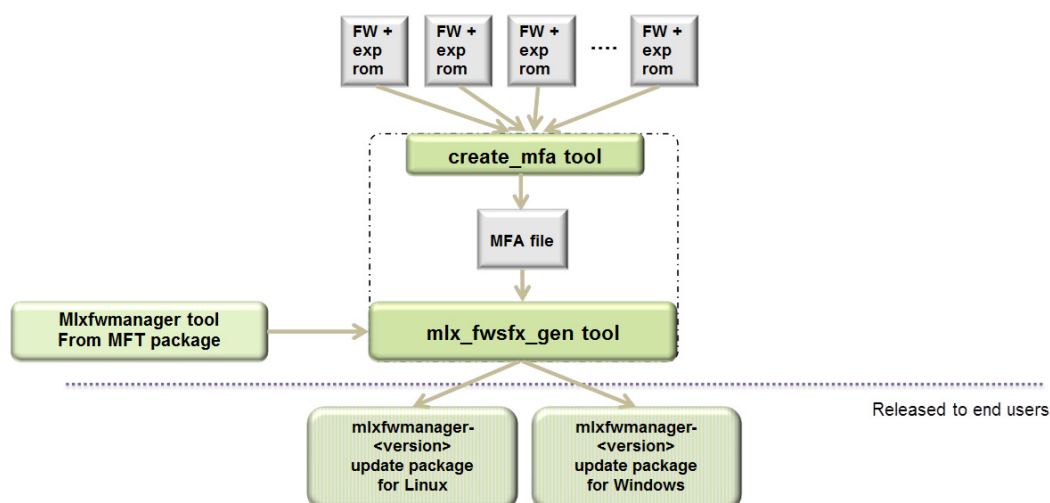
Usage:

```
# 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_fwfsx_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_TD70VMATA_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_fwfsx_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

 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

 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 also 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

- OFED/WinOF driver to be installed and enabled (for ConnectX-3 and ConnectX-3 Pro)
- Access to the device through the PCI interface (pciconf/pci_cr)
- For the adapter cards below, the following firmware versions are required:
 - ConnectX®-3/ConnectX®-3 Pro: v2.31.5000 or above
 - Connect-IB™: v10.10.6000 or above
- Supported devices:
 - Adapter cards: ConnectX®-3 /ConnectX®-3 Pro, Connect-IB®, ConnectX®-4/ConnectX®-4 Lx/ConnectX®-5/ConnectX®-5 Ex/ConnectX®-6/ConnectX®-6 Dx/ConnectX®-6 Lx/NVIDIA BlueField/NVIDIA BlueField-2
 - Switches: Switch-IB/Switch-IB 2,/Spectrum/Spectrum-2/Spectrum-3/Quantum
- Changing device configurations enabled.

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

mlxconfig Synopsis

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

where:

-a --all_attrs	Show all attributes in the XML template
-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
-y --yes	Answers yes in prompt
-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.
-v --version	Displays version info
-h --help	Displays help message
-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
q[query]	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.
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)
i[show_confs]	Display information about all configurations
clear_semaphore	Clear the tool's semaphore
r[eset]	Resets configurations to their default value
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.
-t --device_type <switch/hca>	Specify the device type.
-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.
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.

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)

```

⚠ N/A means that the device default configuration is set.

- ⚠ 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)
```

Using mlxconfig

Using mlxconfig 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_DESCRIPTOR 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_DESCRIPTOR 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 VPI Parameters

In order to set VPI 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 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 QSFP 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 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.



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
```

⚠ 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.

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_pciconfl 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.

```


 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.

 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 0x11250000
# 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>
```

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.

 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.

 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%56%78" --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

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

⚠ 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.

⚠ In case there were no server booting after enabling SR-IOV, please refer to [Troubleshooting](#).

⚠ 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...]
```

Switches 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_f lashes>	Use the given parameters to access the flash instead of reading them from the flash. Supported parameters: <ul style="list-style-type: none"> 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 -> port1 guid+2 -> port2 guid+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, respectively: 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[image] <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
--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
--public_key_label <string>	Flag to use with sign/import_hsm_key commands.
-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/mac) 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.

Note 1. The -mac and -macs options are applicable only to NVIDIA® Ethernet adapter and switch devices.

Note 2. When accessing SwitchX via I2C or PCI, the -override_cache_replacement flag must be set.

Command Parameters

The flint utility commands are:

Common FW Update and Query

b[urn]	Burn flash
q[uey] [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_h mac	Sign image with HMAC.

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 FW Image operations:

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


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


HW Access Key:

set_key [key]	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.
hw_access <enable disable> [key]	Enable/disable the access to the HW. The key can be provided in the command line or interactively typed after the command is given.

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

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

 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)	<p>Optional, for InfiniBand adapters and 4th generation (Group I) switches. One or four GUIDs.</p> <ul style="list-style-type: none">• If 4 GUIDS are provided (-guids flag), they will be assigned as node, Port 1, Port 2 and system 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. <p>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.</p> <p>NOTE: A GUID is a 16-digit hexadecimal number. If less than 16 digits are provided, leading zeros will be inserted.</p>
MAC(s)	<p>Optional, for Ethernet and VPI adapters and switches.</p> <ul style="list-style-type: none">• 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. <p>NOTE: A MAC is a 12-digit hexadecimal number. If less than 12 digits are provided, leading zeros will be inserted.</p>

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/mt4099_pci_cr0 -i fw-4099-2_42_5000-MCX354A-FCB_A2.bin burn
```

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

```
# flint -d /dev/mst/mt4099_pci_cr0 -i fw-4099-2_42_5000-MCX354A-FCB_A2.bin -guid 1234567deadbeef burn
```

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

```
# flint -d /dev/mst/mt4099_pci_cr0 -i fw-4099-2_42_5000-MCX354A-FCB_A2.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/mt4099_pci_cr0 -i fw-4099-2_42_5000-MCX354A-FCB_A2.bin -nofs -guid 12345678 burn
```

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

```
# flint -d /dev/mst/mt4099_pci_cr0 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.

- ⚠ 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.

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

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

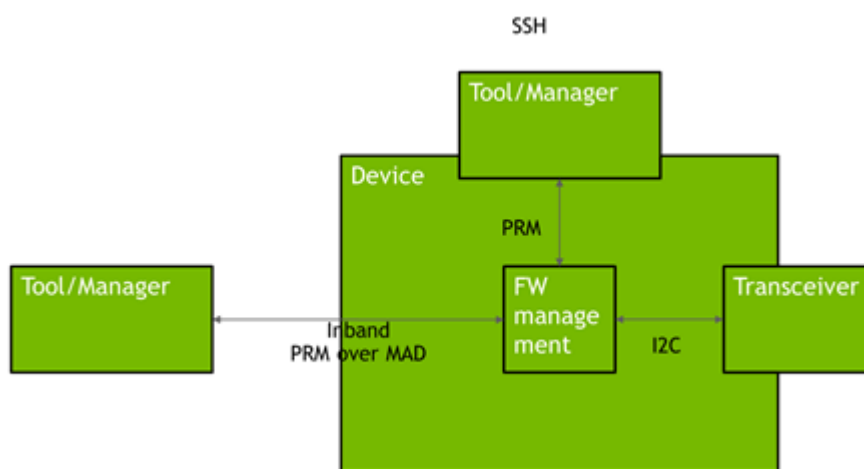
The In-Field-Firmware-Update (IFFU) tool works via the switches/NICs in the datacenters and is intended for remote control. The tool is used to update cables transceivers' firmware.

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.



⚠ 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[uey]	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
```



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
```



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.

⚠ 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.
```

⚠ 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.

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/mt4099_pci_cr0 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/mt4099_pci_cr0 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.

 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)
```

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 -l- Calculating Checksum ... Checksum: 68ddae6bfe42f87f09084f3f468a35c6

```
flint -d /dev/mst/mt4117_pciconf0 cs
-l- 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 guides) 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-burning 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
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

```

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

2. Change GUIDS for device (specifying guides_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 guides_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).

⚠ 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 -wimage./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:          ffffffff ffffffff ffffffff ffffffff
MACs:           ffffffff ffffffff ffffffff ffffffff
```

```
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 0x0002c9bdef1 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:
GUIDs:           Node      Port1      Port2      Sys image
                0002c9030abcdef0 0002c9030abcdef1 0002c9030abcdef2 0002c9030abcdef3
MACs:           0002c9bdef1      0002c9bdef2
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 -wimage ./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 0x0002c9bdef1 -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:
GUIDs:           Node      Port1      Port2      Sys image
                0002c9030abcdef0 0002c9030abcdef1 0002c9030abcdef2 0002c9030abcdef3
MACs:           0002c9bdef1      0002c9bdef2
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.


 The secure host feature requires a MLNX_OFED driver installed on the machine.

4th Generation Devices

To disable/enable access to the hardware:

1. Set the key:

```
# flint -d /dev/mst/mt4099_pci_cr0 set_key 22062011
Setting the HW Key - OK
Restoring signature - OK
```

 A driver restart is required to activate the new key.

2. Access the HW while HW access is disabled:

```
# flint -d /dev/mst/mt4099_pci_cr0 q
E- Cannot open /dev/mst/mt4099_pci_cr0: HW access is disabled on the device.
E- Run "flint -d /dev/mst/mt4099_pci_cr0 hw_access enable" in order to enable HW access.
```

3. Enable HW access:

```
# flint -d /dev/mst/mt4099_pci_cr0 hw_access enable
Enter Key: *****
```

4. Disable HW access:

```
# flint -d /dev/mst/mt4099_pci_cr0 hw_access disable
```



WARNING:

1. Once a hardware access key is set, the hardware can be accessed only after the correct key is provided.
2. If a key is lost, there is no way to recover it using the tool. The only way to recover from a lost key is to:
 - Connect the flash-not-present jumper on the card
 - Boot in "flash recovery" mode
 - Re-burn FW
 - Re-set the HW access key

For further details, please refer to [Secure Host](#).

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.

 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]
```

 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 CONFO 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).



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](#).



- 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

Secure 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

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

```
# mlxburn -fw ./fw-4099-rel.mlx -conf ./secure_host.ini -wimage fw-4099.secure.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

To set the key, run:

```
# flint -d /dev/mst/mt4099_pci_cr0 set_key 22062011
Setting the HW Key - OK
Restoring signature - OK
```

 A driver restart is required to activate the new key.

Disabling/Enabling Access to the Hardware

1. Access the hardware while hardware access is disabled:

```
# flint -d /dev/mst/mt4099_pci_cr0 q
E- Cannot open /dev/mst/mt4099_pci_cr0: HW access is disabled on the device.
E- Run "flint -d /dev/mst/mt4099_pci_cr0 hw_access enable" in order to enable HW access.
```

2. Enable hardware access:

```
# flint -d /dev/mst/mt4099_pci_cr0 hw_access enable
Enter Key: *****
```

3. Disable hardware access:

```
# flint -d /dev/mst/mt4099_pci_cr0 hw_access disable
```

Removing the Secure Host

 This section is applicable to Group 1 network adapters only.

To remove the secure host feature:

1. Make sure you have INI and MLX file suitable for the device.
 - a. Remove `cr_protection_en=true` from the INI (if present)
 - b. Generate the image using `mlxburn`, for example run:

```
# mlxburn -fw ./fw-4099-rel.mlx -conf ./unsecure_host.ini -wimage 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

 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	
	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	Not supported in Secure FW	Not supported in Secure FW	Flash GW is blocked

Tool	Flow	Secure FW	With CS Token	Blocked Commands
	(Direct flash GW access)			
	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
mlxdump	fsdump	Blocked icmds	Working	gcif_get_ft_info, gcif_get_ft_list, gcif_get_fg, gcif_get_fg_list, gcif_get_fte, gcif_get_fte_list
	phyUc	Blocked icmds	working	gcif_phy_uc_get_array_prop_px, gcif_phy_uc_set_get_data, gcif_phy_uc_get_array_prop_EDR, gcif_phy_uc_get_array_prop_HDR
	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_buffer
wqdump	Dump QP contexts	Blocked icmds	working	gcif_read_context
	Dump WQs	Blocked icmds	working	gcif_read_host_memory, gcif_read_qentry, gcif_qp_get_pi_ci
	ICM	Blocked icmds	working	gcif_read_icm
	WRITE QP (Devmon)		working	gcif_write_context
mgettemp	hw_access	Read Only CR- Space	working	Read Only CR- Space

Tool	Flow	Secure FW	With CS Token	Blocked Commands
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
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/ mxlxi2c	Not relevant when not in livefish			
	With Force flag (ENV VAR)	Read Only CR- Space	working	Read Only CR- Space

The following sections describe how Secure FW updates are performed.

Signing Binary Image Files

For firmware Secure purposes, you may sign the image file using the sign command. If you do not provide the sign command with a private key and UUID, the command will only compute SHA256 digest and add it to the image signature section. The sign command supports RSA keys with lengths of 2048 and 4096 bits.

- If you provide a private key with the length of 2048 bits, the command will compute SHA256 digest and encrypt it with the private key and add the result with the provided UUID to the appropriate image signature section.
- If you provide a private key with the length of 4096 bits, the command will compute SHA512 digest and encrypt it with the provided key and add the result with the provided UUID to the appropriate image signature.

You can sign with two keys in the same command by providing keys with lengths of 2048 and 4096 bits. The flags to be used for the first private key and uuid are “--private_key” and “--key_uuid”, and for the second private and uuid use “--private_key2” and “--key_uuid2”.

The motivation for signing with two keys is to allow a firmware update from both firmwares, the one that supports only 2048bit keys and the one that supports 4096bit keys.


Examples:

```
# flint -i /tmp/image.bin sign --private_key privatekey.pem --key_uuid "e0129552-13ba-11e7-a990-0cc47a6d39d2"
```

```
# flint -i /tmp/image.bin sign --private_key privatekey_2048.pem --key_uuid "e0129552-13ba-11e7-a990-0cc47a6d39d2"
--private_key2 privatekey_4096.pem --key_uuid2 "a0b43568-17cb-16e9-a990-0ff47a6d39e4"
```

Hardware Security Module (HSM)

Hardware Security Module (HSM) can be used with flint as well through OpenSSL dynamic engines to allow the user to sign the firmware with a secure key management system instead of directly providing key files to flint.

 HSM and OpenSSL engines should be configured properly to work with dynamic engines.

Example:

```
# flint -i /tmp/image.bin --openssl_engine pkcs11 --openssl_key_id
"pkcs11:serial=0123456789abcdef;token=My%20token%201;type=private;object=example_pkey;id=%12%34%56%78" --key_uuid
"e0129552-13ba-11e7-a990-0cc47a6d39d2"
sign
```

Setting a "Public Keys" Section in a Binary Image File

To override the public keys section in a given binary image file, use `set_public_key`.

```
# flint -i /tmp/image.bin set_public_keys public_key.bin
```

Setting a "Forbidden Versions" Section in a Binary Image File

To override the forbidden versions section in a given binary image file, use `set_forbidden_versions`.


```
# flint -i /tmp/image.bin set_forbidden_versions forbidden_versions.bin
```

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                               GuidNumber
Base GUID:       7cfe90030029205e                  4
Base MAC:        00007cfe9029205e                  4
Image VSD:
Device VSD:
PSID:            MT_2190110032
Security Attributes: secure-fw, dev
```

 Unavailable information is reported as N/A.

In secure firmware, a firmware update will be successful if an image is signed with a valid key that is recognized by the running firmware on the chip. For more information, please refer to [Signing Binary Image Files](#). If the security type permits legacy flash access commands, the `--no_fw_ctrl` flag can be used to command the flint to work in the non firmware controlled mode. This means that all the non-secure functionality will be supported using this flag, and the burn flow will work without requiring a signed image. Example:

```
# flint -d /dev/mst/mt4115_pciconf0 --no_fw_ctrl q
Image type:          F33
FW Version:          12.19.2096
FW Release Date:     26.3.2017
Description:         UID                               GUIDsNumber
Base GUID:           248a07030094050c                  4
Base MAC:            0000248a0794050c                  4
Image VSD:
Device VSD:
PSID:                MT_2170110021
```

Re-Signing a Binary Image File

The following procedure is intended to be implemented by customers who want to use their keys to sign a secured firmware.

1. Set the public keys in a given firmware image:
 - a. Generate a binary file that contains 8 public keys.

You can use `mlxconfig` command `xml2bin` to generate the file:

 - i. To generate 2048 bits public keys:
 1. Run: `mlxconfig gen_tlvs_file output.txt`.
 2. Open the `output.txt`.
 3. Go to the line starting with "file_public_key" and change the 0 to 1.
 4. Save the file and exit.
 5. Run: `mlxconfig gen_xml_template output.txt output.xml`
 6. Open the `output.xml`.
 7. Duplicate the xml node "file_public_key" so the file has 8 copies, for each node fill it as follows:
 - `cs_token_en` = 0
 - `fw_en` = 1
 - `mlnx_nvconf_en` = 1
 - `vendor_nvconf_en` = 1
 - `auth_type`: 0x3 for 2048 bits keys and 0x4 for 4096 bits keys.

Example for `public_key_exp`, `keypair_uuid`, `key`:

[illegible]

You can have spaces between the bytes: `f8 00 00 03`, or you can have multiple lines.


The order of the bytes is the same as the output of openssl file, Therefore, you can take the key as is from the openssl file.

8. Save and Exit.
9. Run: `mlxconfig xml2bin output.xml output.bin`.


- ii. To generate 4096 bits public keys, please follow the same steps as above, but use "file_public_key_4096" instead of "file_public_key".
For further information, see [mlxconfig xml2bin Command](#).
 - b. Set the key's binary file in the firmware image using the flint set_public_keys command.
For further information, see [Setting a "Public Keys" Section in a Binary Image File](#).
2. If there is need to modify the definition for the forbidden_versions in a given firmware image then:
 - a. Generate a binary file that contains the forbidden versions.
You can use the mlxconfig command xml2bin to generate it according to the steps described in Step a above (Generate a binary file that contains 8 public keys).
An example for forbidden versions xml node:


```
<nv_forbidden_versions>
<creation_time_day>18</creation_time_day>
<creation_time_month>6</creation_time_month>
<creation_time_year>7e2</creation_time_year>
<creation_time_second>d</creation_time_second>
<creation_time_minute>19</creation_time_minute>
<creation_time_hour>12</creation_time_hour>
<min_allowed_fw_version>0</min_allowed_fw_version>
<forbidden_fw_version index="0">53:1f:0d06</forbidden_fw_version>
<forbidden_fw_version index="1..31">0</forbidden_fw_version>
</nv_forbidden_versions>
```

- b. Set the key's binary file in the firmware image using the flint set_forbidden_versions command.
For further information, see [Setting a "Forbidden Versions" Section in a Binary Image File](#).
3. Sign the firmware image with a private key.

 Please notice that signing the image must be after setting the public keys, and the forbidden versions. For further information, see [securefmupdate](#).

- a. Run the flint sign command.

 To sign with a 2048 bits private key only, make sure that the firmware image does not contain a 4096 bits key signature.
Run the flint set_public_keys command with a 4096 bits keys section filled with zeros and then sign with a 2048 bits private key.
For further information, see [Signing Binary Image Files](#).

 To sign with 4096 bits private key only, run the flint set_public_keys command with a 2048 bits keys section filled with zeros and then sign with the a 4096 bits private key.
For further information, see [Signing Binary Image Files](#).

Secure Boot

The Secure Boot is a mechanism used to ensure the integrity of the running firmware on a device. After Secure Boot is enabled, only firmware signed with an approved keys and appropriate security-version can be executed. The Secure Boot mechanism relies on public/private key pairs to verify the digital signature of all firmware before execution.

 Secure Boot is supported only on ConnectX-6 Dx onwards adapter cards.

For Secure Boot purposes, you should sign the image file using the `rsa_sign` command. The command supports only keys with the length of 4096 bits.

Example:

```
# flint -i /tmp/image.bin --private_key privatekey.pem --public_key publickey.pub --key_uuid "e0129552-13ba-11e7-a990-0cc47a6d39d2" rsa_sign
```

Using HSM:

```
# flint -i /tmp/image.bin --openssl_engine pkcs11 --openssl_key_id "pkcs11:serial=0123456789abcdef;token=My%20token%201;type=private;object=example_pkey;id=%12%34%56%78" --key_uuid "e0129552-13ba-11e7-a990-0cc47a6d39d2" --public_key publickey.pub rsa_sign
```

mlxburn - Firmware Image Generator and Burner

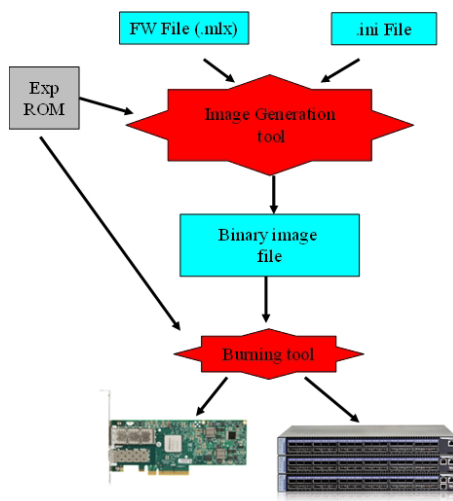
mlxburn is a tool for firmware (FW) image generation and/or for burning a firmware image to the Flash/EEPROM attached to an NVIDIA® device. Both functions or a single function of mlxburn can be activated by means of command line options (see [mlxburn Synopsis](#)). It can also query for firmware attributes (e.g., firmware version, GUIDs, etc.) and VPD info of adapter cards and switch systems.

mlxburn allows for customization of standard NVIDIA® firmware for OEM specific needs (e.g., a specific adapter board type). See [Customizing Firmware](#).

Generating and Burning Firmware

The mlxburn firmware update flow is composed of two separate stages: image generation and image burning. In the image generation stage, a given NVIDIA® firmware release in .mlx format is processed together with a board-specific configuration (.ini) file to generate a ‘burnable’ firmware image. This image is burnt to the Flash/EEPROM attached to an NVIDIA® device in the second stage. The burning process retains device specific data such as GUIDs, UUIDs, MACs, VSD, and BSN. Also, the burn process is failsafe by default.

FW Generation and Burning



mlxburn runs both stages by default, but it may perform only one by means of command options. If ‘-wimage’ is specified (see [mlxburn Synopsis](#)), only image generation is performed. Specifying the

‘-image’ option skips the image generation stage and loads the provided image (generated in a previous run of mlxburn using the ‘-wrmimage’ option).

Customizing Firmware

An NVIDIA® firmware image can be customized (usually) to fit a specific board type. The customization is done by using a FW parameter-set file in the image generation stage. This file has a .ini format. Each parameter-set file has a unique parameter-set ID (PSID), which is kept in the device Flash/EEPROM and allows retaining device configuration during future FW updates.

During a device FW update, mlxburn reads the PSID from the device and uses the corresponding .ini file when generating the FW image. mlxburn searches for the files in the same directory of the FW release. When mlxburn is used to generate an image file, or when no corresponding parameter-set file is found, the user should explicitly specify which parameter-set file to use.

To produce an image file the user needs to provide the option ‘-wrmimage <target file>’. To actually burn the image to the Flash/EEPROM attached to an NVIDIA® adapter or switch device, the user needs to specify the option ‘-dev <mst device>’ (see the synopsis section below).

If run in burning mode, mlxburn auto-detects the firmware parameter-set with which the device was previously burnt. It locates and uses this parameter-set file to generate the appropriate image for the device (by merging the FW release with the specific parameter-set required).

To inhibit image generation, the ‘-image <pre-generated-image-file>’ should be used. It instructs mlxburn to use the given file for burning the device.

mlxburn Synopsis

```
#mlxburn [-h][-v] <-dev mst-device|-wrmimage fw-image>
<-fw mellanox-fw-file|-image fw-image|-img_dir img_directory|-fw_dir fw_dir> [-conf fw-conf-file] [-nofs] [-nofs_img]
[-striped_image] [-format BINARY|IMAGE] [-dev_type device type] [-exp_rom <exp_rom_file>] [-exp_rom_dir <exp_rom_dir>]
[-force] [-conf_dir <conf_dir>] [-gb_bin_file <gb_bin_file>] [-fwver] [-vpd] [-vpd_rw] [-vpd_prog_rw <rw-keywords-
file>] [-vpd_set_keyword <keyword-assignment>] [-set_pxe_en <(port1|port2)=(enable|disable)>] [-prof_file <profiles
file>]
[-query] [-conf_dir_list <dir1,dir2,...,dirn>]
```

Note: The “-fwver” flag is not supported in Connect-IB®, Switch-IB™, ConnectX-4® and ConnectX-5® devices.

where:

-dev_type <mellanox-device-number>	<p>mlxburn must know the device type in order to work properly. Use this flag if device type auto-detection fails. Example: -dev_type 23108</p> <p>Supported NVIDIA® device types:</p> <ul style="list-style-type: none"> HCAs/NICs: 25408, 25418, 26418, 26428, 25448, 26448, 26468, 26478, 25458, 26458, 26438, 26488, 4099, 4103, 4113, 4115, 4117, 4119, 4121, 41680, 41681, 41682. Switches: 48436, 48437, 48438, 51000, 52000, 52100, 53000, 53100.
-fw <mellanox-fw-file>	Specify NVIDIA® FW released Firmware File to use (file extension is .mlx)
-conf <parameter-set-file>	FW configuration file (.ini). Needed when generating image (not using -dev flag) or if configuration auto detection fails.

-conf_dir <dir>	When specified, the auto detected configuration files will be looked for in the given directory, instead of in the firmware file directory. Applicable for burn operation.
-gb_bin_file <gb_bin_file>	Integrate the given gearbox binary file to the FW image.
-dev <mst-dev>	Burn the image using the given mst device
-exp_rom <exp-rom-file>	<p>Integrate the given expansion rom file to the FW image. The given file may be in .img or bin/.rom (raw binary) format.</p> <ul style="list-style-type: none"> If the exp-rom-file is set to "AUTO", expansion rom file is auto detected from the files rom in the exp_rom_dir (see below). <p>NOTE: Exp rom auto detection is done for devices that are already burned with an exp-rom image.</p> <ul style="list-style-type: none"> If "-exp_rom AUTO" is specified for a device with no exp-rom, it would be burnt with no exp rom. <p>To add exp-rom to a device, manually supply the exp rom file to use.</p>
-exp_rom_dir <exp_rom_dir>	The directory in which to look for expansion rom file when "-exp_rom AUTO" is specified. By default, exp-rom files are searched in <fw file directory>/exp_rom/*
-force	None interactive mode. Assume "yes" for all user questions.
-format <BINARY IMAGE>	Specify which image format to use. Can be specified only with the -wimage flag. Default is BINARY.
-fw_dir <dir>	When specified, the auto detected fw files will be looked for in the given directory. Applicable for burn operation.
-conf_dir_list <dir1,dir2,...,dirn>	When specified, the auto detected configuration files will be looked for in the given directories, instead of in the firmware file directory. Applicable for burn operation.
-fwver	<p>When a device is given: Display current loaded firmware version.</p> <p>When a FW file is given (-fw flag): Display the file FW version.</p> <p>Note: The "-fwver" flag is not supported in Connect-IB® devices.</p>
-h	Display a short help text
-image <fw-image-file>	Do not generate image. Use the given fw image instead
-img_dir <image directory>	Do not generate image. Select the image to burn from the *.bin in the given directory
-nofs	When specified, burn process will not be failsafe.
-nofs_img	When specified, generated image will not be failsafe, and burn process will not be failsafe
-striped_image	When specified, generated image will be in striped format, and will indicate that the image is in striped format when queried.
-query	<p>Query the HCA/Switch device for firmware details, e.g. Firmware Version, GUIDs etc.</p> <p>In addition to the above flags, Mlxburn can also accept the following flags/options, which are passed to the underlying burning tool:</p> <pre>-banks -use_image_ps -skip_is -mac(s) -guid(s) -sysguid -vsd -ndesc -bsn -pe_i2c -se_i2c -is3_i2c</pre>

	-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_prog_rw<rw-keywords-file> _{1,2}	<p>(on Linux only): Program the VPD-W tag (the writable section of the VPD) with the data given in the rw-keywords-file. File lines format: "KEYWORD = VALUE".</p> <p>In order to set binary data to a keyword, add ":BIN" to the keyword name. in this case, the data is a hexadecimal string of even length. Example file: V1 = MY-ASCII-KEYWORD V2:BIN = 1234abcd</p> <p>White spaces before and after VALUE are trimmed.</p>
-vpd_rw _{1,2}	(on Linux only): Display also the read/write section of the PCI VPD of the given device.
-vpd_set_keyword <keywordassignment> _{1,2}	Add or change a keyword value in the VPD-W tag (the writable section of the VPD) with the data given in the keyword-assignment string. The string format is identical to a line in the rw-keywordsfile described above. Other keywords in the VPD-W tag are not affected by this operation.
-wrimage <fw-image-file>	Write the image to the given file.

eNote 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
```



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

Examples of mlxburn Usage

Host Channel Adapter Examples

To update firmware on an MT25408 ConnectX adapter device with the configuration file (.ini) auto-detected, enter:

```
# mlxburn -fw ./fw-ConnectX3-rel.mlx -dev /dev/mst/mt4099_pci_cr0
```

To generate a failsafe image file for the same adapter above without burning, enter:

```
# mlxburn -fw ./fw-ConnectX3-rel.mlx -conf ./MCX354A-FCB_A2-A5.ini -wimage ./fw-4099.bin
```

To update firmware on the same adapter above with the configuration file (.ini) explicitly specified, enter:

```
# mlxburn -fw ./fw-ConnectX3-rel.mlx -dev /dev/mst/mt4099_pci_cr0 -conf ./CX354A-FCB_A2- A5.ini
```

ConnectX-5 Examples

To generate a failsafe image file for ConnectX-5® device without burning, enter:

```
# mlxburn -fw FW/fw-ConnectX-5.mlx -conf FW/CX515A-CCA_Ax.ini -wimage fw-ConnectX-5-CX515A-CCA_Ax.bin -base_guid 0x002c90330123e00
```

To update firmware on a ConnectX-5® device, enter:

```
# mlxburn -i fw-ConnectX-5-CX515A-CCA_Ax.bin -d /dev/mst/mt4115_pciconf0
```

ConnectX-4 Examples

To generate a failsafe image file for ConnectX-4® device without burning, enter:

```
# mlxburn -fw FW/fw-ConnectX-4.mlx -conf FW/MCX456A-ECA_Ax.ini -wimage fw-ConnectX-4-MCX- 456A-ECA_Ax.bin -base_guid 0xe002c903002ef500
```

To update firmware on a ConnectX-4 device, enter:

```
# mlxburn -i fw-ConnectX-4-MCX456A-ECA_Ax.bin -d /dev/mst/mt4113_pciconf0
```

ConnectX-4 Lx Examples

To generate a failsafe image file for ConnectX®-4 Lx device without burning, enter:

```
# mlxburn -fw FW/fw-ConnectX4Lx.mlx -conf FW/MCX4131A-GCA_Ax.ini -wimage fw-ConnectX-4LX- MCX4131A-GCA_Ax.bim -base_guid 0xe41d2d0300ab2a4e -base_mac 0000e41d2dab2a4e
```

To update firmware on a ConnectX-4 Lx device, enter:

```
# mlxburn -i fw-ConnectX-4LX-MCX4131A-GCA_Ax.bim -d /dev/mst/mt4117_pciconf0
```

Connect-IB Examples

To generate a failsafe image file for Connect-IB® device without burning, enter:

```
# mlxburn -fw FW/fw-ConnectIB.mlx -c FW/MCB194A-FCA_A1.ini -wimage fw-ConnectIB-MCB194A-FCA_A1.bin -base_guid 0x0002c903002ef500
```

To update firmware on a Connect-IB® device, enter:

```
# mlxburn -i fw-ConnectIB-MCB194A-FCA_A1.bin -d /dev/mst/mt4113_pciconf0
```

SwitchX Switch Examples

Burn an MT51000 switch system using the In-Band access method:

```
# mlxburn -dev /dev/mst/SW_MT51000_000002c900002100_lid-0x000E -fw ./fw-sx.mlx
```

Generate an MT51000 image and perform an In-Band update of the device with LID 0xE:

```
# mlxburn -dev lid-0x000E -fw ./fw-sx.mlx
```

Generate and burn a new MT51000 via I2C:
Set the I2C network to access the SwitchX switch.

```
# mlx_i2c -d /dev/mst/mtusb-1 p SX
```

Burn the new image (the flash is still blank) specifying the Node GUID, system GUID, base MAC and Switch MAC. Note that 4 guids (in quotes) should be specified as an argument to the -guids flag. The 2 middle GUIDs are ignored by SwitchX and should be set to 0.

```
# mlxburn -d /dev/mst/mtusb-1 -fw ./fw-sx.mlx -conf MSX6025F_A1.ini -guids "000002c900002100 0 0 000002c900002100" -macs "0002c9002100 0002c9002101" -nofs
```

NVIDIA® Spectrum® Examples

To generate a failsafe image file for a NVIDIA® Spectrum® device without burning, enter:

```
mlxburn -fw FW/fw-SwitchEN.mlx -c FW/MSN2700-Cxxx_Ax.ini -wrmage fw-Spectrum-MSN2700-Cxxx_Ax.bin -base_guid e41d2d030045a240 -base_mac 0000e41d2d45a240
```

To update firmware on a Spectrum™ device, enter:

```
mlxburn -i fw-Spectrum-MSN2700-Cxxx_Ax.bin -d /dev/mst/mt52100_pciconf0
```

Switch-IB Examples

To generate a failsafe image file for a Switch-IB™ device without burning, enter:

```
mlxburn -fw FW/fw-SwitchIB.mlx -c FW/MSB7700-E_Ax.ini -wrmage fw-SwitchIB-MSB7700-E_Ax.bin - base_guid 0x0002c903002ef500
```

To update firmware on a Switch-IB device, enter:

```
mlxburn -i fw-SwitchIB-MSB7700-E_Ax.bin -d /dev/mst/SW_MT52000_000011111101a24c_lid- 0x0006,mlx4_0,1
```

Switch-IB 2 Examples

To generate a failsafe image file for a Switch-IB2™ device without burning, enter:

```
mlxburn -fw FW/fw-SwitchIB-2.mlx -c FW/MSB7800-Exxx_Ax.ini -wrmage fw-SwitchIB-2- MSB7800- Exxx_Ax.bin -base_guid 7cfe900300a5a620
```

To update firmware on a Switch-IB 2 device, enter:

```
mlxburn -i fw-SwitchIB-2- MSB7800-Exxx_Ax.bin -d /dev/mst/mt53000_pciconf0
```

Exit Return Values

The following exit values are returned:

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

mlxfwreset - Loading Firmware on 5th Generation Devices Tool

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

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..5>] [--type <0..1>] [--sync <0..1>] [-s] [-m]
```

mlxfwreset Synopsis

where:


q query	Query supported reset level/type/sync
r reset	Execute reset
reset_fsm_register	Reset the multi-host synchronization register
-d --device <device>	Device to work with
-l --level <0..5>	Run reset with the specified reset-level
-t --type <0,1>	Run reset with the specified reset-type
--sync <0,1>	Run reset with the specified reset-sync
-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" Note: This option is supported in Linux OSes only.
-s --skip_driver	Skip driver start/stop stage (driver must be stopped manually)
-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")
 - 3: Driver restart and PCI reset
 - 4: Warm Reboot
 - 5: Cold Reboot
- Reset-types (relevant only for reset-levels 3, 4):
 - 0: Full chip reset
 - 1: Phy-less reset ("port-alive" - network link will remain up)

 The exact reset-level and reset-types needed in order 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

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.

 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

Running mlxfwreset on a SmartNIC device is identical to running mlxfwreset on a Multi-Host setup while the integrated Arm is considered as one of the hosts.

The procedure on a SmartNIC device is to run mlxfwreset first from the integrated Arm and then from all other hosts.

 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.

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
3: Driver restart and PCI reset -Supported (default)
4: Warm Reboot -Supported
5: Cold Reboot -Supported

Reset-types (relevant only for reset-levels 3,4):
0: Full chip reset -Supported (default)
1: Phy-less reset ("port-alive" - network link will remain up) -Not Supported

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

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

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

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.

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[uey]
```

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[uey]	Query PHY FW on device.



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

 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 at [Mellanox.com](#) or 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

```

⚠ 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*.

⚠ It is recommended to burn the FPGA device using an RDMA device as it is 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:

<code>--factory</code>	Load FPGA image from factory flash
<code>--user</code>	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: 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)

 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 programming 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 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`.

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.


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! |
+++++++
```

mstcongestion - Tool for Setting Congestion Mode and Action

mstcongestion is a tool used to configure device's behavior in case of excessive ingress traffic where the ingress traffic is higher than the PCIe capability. The excessive traffic can either be dropped (drop action) or marked as CE (Congestion Encountered) in the IP header.

The tool can work in either aggressive mode where traffic is dropped/marked in an aggressive way, or in dynamic mode where the drop/mark is more relaxed.

 mstcongestion is not supported in ESXi 7.0.

 mstcongestion is supported on ConnectX-4 Lx onwards Multi-Host devices only.

Tool Requirements

- Firmware version ConnectX-4 Lx: 14.23.1020 or later

mstcongestion Synopsis

```
# mstcongestion [option] [-d|--device <PCI DEVICE>] [--mode <MODE>] [--action <ACTION>] [-q|--query] [-h|--help] [-v|--version]
```

where:

-d --device <PCI DEVICE>	NVIDIA® PCI device address
--mode <MODE>	Set Mode, options are: [aggressive dynamic]
--action <ACTION>	Set Action, options are: [disabled drop mark] Note: The “mark” option is available only if the driver supports such capability.
-q --query	Query congestion
-h --help	Show help message and exit
-v --version	Show version and exit

mlxprivhost - NIC Configuration by the Host Restriction Tool

mlxprivhost enables the user to restrict the hosts from managing the device.

⚠️ mlxprivhost is supported in Linux only.

⚠️ mlxprivhost is not supported in ESXi 7.0.

⚠️ This utility is supported in BlueField devices only.

mlxprivhost Synopsis

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

- Restrict configuration takes effect immediately, but with disabling/enabling RShim, it requires reboot
- The host cannot restrict itself, it only restricts other external hosts
- A restricted host will not be able to perform operations as the below that can compromise the DPU:

- 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 system, the tool is compatible with firmware version 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
--device DEVICE, -d DEVICE	Device to work with.
--disable_rshim	When TRUE, the host does not have an RSHIM function to access the embedded CPU registers. Reboot is required for changes to take effect.
--disable_tracer	When TRUE, the host will not be allowed to own the Tracer
--disable_counter_rd	When TRUE, the host will not be allowed to read Physical port counters
--disable_port_owner	When TRUE, the host will not be allowed to be Port Owner
r,restrict	Set all external hosts restricted except 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 if all external hosts are restricted

Example of mlxprivhost:

- Enabling Full Host Restriction:

```
mlxprivhost -d /dev/mst/mt41682_pciconf0 r --disable_rshim --disable_tracer --disable_counter_rd --disable_port_owner
```

- Disabling Host Restriction:

```
mlxprivhost -d
```

- Query the status of host:

```
mlxprivhost -d
```

Debug Utilities

This section contains:


- [fwtrace Utility](#)
- [itrace Utility](#)
- [mstdump Utility](#)
- [mlx2c Utility](#)
- [i2c Utility](#)
- [mget_temp Utility](#)
- [mlxtrace Utility](#)
- [mldump Utility](#)
- [mlxmcg Utility](#)
- [pckt_drop Utility](#)
- [mlxuptime Utility](#)
- [wqdump Utility](#)
- [mlxmdio Utility](#)
- [mlxreg Utility](#)
- [mlxlink 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.

 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.

 Memory mode on 5th generation (Group II) devices is supported only by PCI mst devices.

 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.32.1xxx).

 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.

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)
-s --stream	Run in streaming mode
-c --cfg	HW tracer events cfg file
-n --snapshot	Take events snapshot - this assumes previous FW configurations
-S --buf_size	HW tracer MEM buffer size in [MB]
--dump	Dump file name
-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
v --version	Print tool's version and exit
--gymi	Global virtual machine interface
--ignore_old_events	Ignore collecting old events

Device Specific Info:

- Connect-IB®, ConnectX®-4, ConnectX®-4 Lx, ConnectX-6 Lx, Switch-IB, Switch-IB 2, Spectrum, Spectrum-2, Quantum:
iRISC names: [i0, iron, i2, i3, i4, i5, i6, i7, all]
- Spectrum-3, Quantum-2:
iRISC names: [i0, iron, i2, i3, i4, i5, i6, i7, i8, i9, all]
- ConnectX®-5, BlueField, ConnectX®-6, ConnectX-6 Dx, BlueField-2
iRISC names: [i0, iron, i2, i3, i4, i5, i6, i7, i8, i9, i10, all]
- Trace classes:
DEBUG_INIT, INIT, ICM, ICM_FREE_LIST, HOST_MNG, CMD_IF, PHY_IB, PHY_RX_ADAP, LIBFHI, PHY_COMMON, PHY_MANAGER, PWR, FLR, ICM_ACCESS, MAD, RXT_CHECKS, I2C, TRANSPORT, FW_LL, RX_ERRORS, CMD_DRIVER, PROFILING, MANAGEMENT, FLASH, STEERING, IFARM, ICMD, PCI, DC_CLEANUP, PHY_ETH, VIRT

Example:

```
# fwtrace -d mlx5_0 -i all -s
-I- Found FW string db cache file, going to use it
mlxtrace -d mlx5_0 -m MEM -c /tmp/itrace_8153.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_8153.cfg
```



```

-I- Output file (Trace File) Path           : mlxtrace.trc
-I- User Buffer Size                       : NA[MBytes]
-I- Use Stream Mode                       : YES
-I- Configure Only                        : NO
-I- Only Snapshot (Skip Configuration Stage) : 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                  : 0[usec]
-I- =====
-I- Device is: cib
-I- Configuring Tracer...
-I- Invalidating kernel buffer... (Press ^C to skip)
-I- Done
-I- Tracer was configured successfully
Device frequency: 276MHz
-I- Starting event streaming...
Reading new events...
774774193803 I2 Mad received on port 1 - QP 0
774774215444 I2 process set_get_pkey_table on port=1 set_get_=0 block=1
774775079296 I2 Mad received on port 1 - QP 0
774775120645 I2 port_state changed from INIT to ARM
774775166315 I2 process set_get_port_info on port 1 set_get_: 1 status:0x0
774775335890 I2 Mad received on port 1 - QP 0
774775367205 I2 port_state changed from ARM to ACTIVE
774775410880 I2 process set_get_port_info on port 1 set_get_: 1 status:0x0
774786733806 I3 process MAD_IFC on port 1
774786744859 I3 process set_get_port_info on port 1 set_get_: 0 status:0x0
.
.
.

```

itrace Utility

The itrace utility extracts and prints trace messages generated by the firmware of a ConnectX-2/ ConnectX-3/ConnectX-3 Pro adapter cards. These trace messages inform developers of software drivers about internal status, events, critical errors, etc., for each iRISC. Trace messages generated by iRISCs are stored in the trace buffer. The trace buffer is located in host memory for MemFree adapter cards (i.e., without on-board memory), and in adapter memory for adapter cards with on-board memory.

The utility is a command line application controlled by command line parameters. It prints trace messages in text format to the console.

itrace Usage

In order to print the firmware traces, the following are required:

- Debug firmware is burnt and loaded on the device
- The driver is up, meaning:
 - For adapters with on-board memory: The SYS_ENABLE command has been executed
 - For adapters without on-board memory (MemFree): The RUN_FW command has been executed
- The desired trace mask is set (see the -m flag below)

The mst driver must be started prior to running itrace tool. To start itrace:

1. Start the mst driver (mst start or mst restart).
2. Enter the following command:

```
# itrace [options...] IRISC_NAME
```

where:

IRISC_NAME	The iRISC for which traces are to be printed. This can be specified once anywhere in the command line as a special option without the leading hyphen. Run 'itrace -h' to get a list of iRISC names for each adapter device.
-h, --help	Displays help about itrace usage.

-m -- mask=TRACE_MAS K	Sets the Trace Mask.
------------------------------	----------------------

To enable generating trace messages for an iRISC, the trace_mask register must be set according to the specifications in the device's *Programmer's Reference Manual*. Setting or clearing bits of the trace_mask register enables or disables, respectively, the generation of specific types of trace messages.

The trace_mask parameter must be either a hexadecimal or a decimal number and its value will be written into the trace_mask register. Changing the trace_mask parameter will not change or remove messages previously stored in the trace buffer, so disabled types of messages can still be displayed by itrace if they were previously generated.

-w, --wait	Runs itrace in wait mode. itrace will exit only if you press <Ctrl-C>. This is not the default behavior of itrace. Without the -w option, itrace will exit if there have been no new traces in the last 0.5 seconds.
-d, --device=DEVICE	Specifies the name of the mst device driver for accessing the cr-space. The default value is: /dev/mst/mt4099_pci_cr0 . To run itrace via the I2C interface, use this option to specify the following: -d=DEVICE, where the device is an I2C device (such as mtusb-1)
-l, --nolock	Ignore NSI GW lock
--nomap	Sets the itrace to not access memory directly (via memory mapping) for reading the trace buffer, but to use the adapter memory access Gateway instead. By default, itrace accesses the memory directly. If the cr-space device specified by the -d parameter is one of the I2C devices, -nomap is switched on.
--no-propel	Sets the itrace not to animate the propeller in wait mode (-w option). By default, animation is enabled.
-v, --version	Prints the MFT version and exits.
-c, --color	Enables color in trace output.
-D, --dump	Dumps the trace buffer and exits. This option is useful for debugging itrace; it dumps the contents of the trace buffer in row format.
--start=START_NO or --start=now	Sets first message number to display.
--debug=TSTRING	Control trace. See: --help-debug.
--help-debug	Prints trace usage.



For Linux, device names should be listed with the /dev/mst prefix. For Windows, no prefix is required.

Example:

```
itrace -d /dev/mst/mt4099_pci_cr0 sxl
itrace: read memory (174712 bytes), each dot denotes 2048 bytes:
[.....]
IRISC Trace Viewer (Mellanox ConnectX), mft 4.1.0-26, built on Aug 16 2015, 17:32:24. Git SHA
Hash: dd3f359
FW Version: 2.34.8420 09/08/2015 19:44:8
(00000003 c1b59e4e) SCHD: exeqpc_valid2freed(0x0) vec_busy_valid=0x00000010
(00000004 dda895e4) SCHD: SQP:0x000400 exes_super_scheduler:busy_done
```

```
(00000005 dda89760) SCHD: writing QpState SQPSTATE_GOOD_IDLE!!!!
(00000006 dda89868) SCHD: exeqpc_valid2freed(0x0) vec_busy_valid=0x00000010
(00000007 dda97ccf) SCHD: SQP:0x000400 exes_super_scheduler:busy_
(00000008 dda97e47) SCHD: writing QpState SQPSTATE_GOOD_IDLE!!!!
(00000009 dda97f4f) SCHD: exeqpc_valid2freed(0x0) vec_busy_valid=0x00000010
(0000000a dda9a8f6) SCHD: SQP:0x000400 exes_super_scheduler:busy
(0000000b dda9aa6e) SCHD: writing QpState SQPSTATE_GOOD_IDLE!!!!
(0000000c dda9ab79) SCHD: exeqpc_valid2freed(0x0) vec_busy_valid=0x00000010
(0000000d ddaaadcd1) SCHD: SQP:0x000400 exes_super_scheduler:busy_
(00000029 ddaee521) INFO: IPCdata[00]=0x01abcd0a(0000002a ddaee60c) INFO: IPCdata[01]=0x00000014
(0000002b ddaee8ce) MAD: exes_mad: QPN=0x000000, nda_nds=0x7c58d014
(0000002c ddaee9f2) SCHD: SQP:0x000000 sqpc_access_db_algorithm: INC
(0000002d ddaef0d5) SCHD: exes_scheduler: try to insert
(0000002e ddaef2d9) SCHD: SQP:0x000000 exes_scheduler chosen
(0000002f ddaef6aa) SCHD: EXES_GO(0x0)..
```

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.

⚠ 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.32.1xxx).

⚠ 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>
```

where the -full flag dumps all internal registers.

Example:

```
[root@mymach]# mstdump /dev/mst/mt4099_pci_cr0 > mt4099.dmp
```

This dumps the internal configuration data of the device into the file mt4099.dmp.

mlx2c Utility

The mlx2c utility provides a way to route the I2C bus to 4th generation (Group I) switches.

mlx2c Usage

The mst driver must be started prior to running mlx2c.

To start mlx2c:

1. Start the mst driver (mst start or mst restart). Note: This step is not required in Windows.
2. Run mlx2c with the following command line syntax:

```
# mlx2c [switches...] <command> [parameters...]
```

Switches Options

-d <device>	mst i2c device name default: "/dev/mst/mtusb-1" Affected commands: all
-h	Print this help information
-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:

```
# mlx-i2c -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 is 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_width> = 0.
<data>	Data (bytes of length data_width) to write to target device.

 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.

Example on how to read a device temperature:

```
# mget_temp -d /dev/mst/SW_MT51000_0002c903007e76a0_lid-0x0002
```

⚠ mget_temp utility reads the IC temperature, it does not support reading temperature from peripheral sensors on the board.

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.

⚠ 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.

⚠ 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.32.1xxx).

⚠ 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.

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, MEM, DMEM, FMEM, VMEM (possible values="OB_GW", "MEM", "DMEM", "FMEM", "VMEM")
-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 cyclicly (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)
--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/mt4099_pci_cr0 -c connectx3.cfg -m MEM -o connectx3.trc
```

To generate a .trc file with a maximal size of 100 MB, run the following command:

```
# mlxtrace -d /dev/mst/mt4099_pci_cr0 -c connectx3.cfg -m MEM -s 100 -o connectx3.trc
```

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

 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.

- 
- 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:

```
# pkt_drop -d /dev/mst/mt4117_pciconf0 -p 1
```

The example above shows how to use the pkt_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.ussec]
```

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.

 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, Ege, Connect-X3/Pro: MCG, 5th generation devices: MKC, SXDC, FullQp ConnectX-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 - 0xffffffff):
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 - 0xffffffff):
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: <ul style="list-style-type: none"> • 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.

 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
```

 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
```

 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.

 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

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
PTVS
ROCE_ACCL
SBDM
SBDCR
SBDM
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 issues related to them. The tool can be used on different links and cables (passive, active, transceiver and backplane).



- 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.
- Do not use mlxlink to disable the port connecting between the host and the unmanaged switch using (“--port_state dn”) flag.
- mlxlink errors, warnings and notes are printed on stderr console.
- Setting the speeds (50GbE, 100GbE and 200GbE) for the new devices (NVIDIA Connect-X 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]`.

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.

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)/PH(phy loopback)/EX(external loopback)]
-k --fec <fec_override>	Configure FEC [AU(Auto)/NF(No-FEC)/FC(FireCode FEC)/RS(RS-FEC)]
--fec_speed <fec_speed>	Speed to Configure FEC [100G/50G/25G/...] (Default is Active Speed)
--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 Permanently (Optional)
--tx_group_map <group_num>	Map ports to group <group_num> (for NVIDIA Spectrum-2 and NVIDIA Quantum devices)
--ports <ports>	Ports to be mapped [1,2,3,4..]
--test_mode <prbs_mode>	Physical Test Mode Configuration [EN(enable)/DS(disable)/TU(perform tuning)]
--rx_prbs <rx_prbs_mode>	RX PRBS Mode [PRBS31(Default)/PRBS7/...] (Optional - Default PRBS31)

--tx_prbs <tx_prbs_mode>	TX PRBS Mode [PRBS31(Default)/PRBS7/...] (Optional - Default PRBS31)
--rx_rate <rx_lane_rate>	RX Lane Rate [EDR(Default)/25G/10G/...] (Optional - Default 25G)
--tx_rate <tx_lane_rate>	TX Lane Rate [EDR(Default)/25G/10G/...] (Optional - Default 25G)
--invert_tx_polarity	PRBS TX polarity inversion (Optional - Default No Inversion)
--invert_rx_polarity	PRBS RX polarity inversion (Optional - Default No Inversion)
--lanes	PRBS lanes to set (one or more lane separated by comma) [0,1,2,...] Optional: Default all lanes
-b --ber_collect <csv_file>	Port Extended Information Collection [CSV File]
--amber_collect <csv_file>	AmBER Port Extended Information Collection For 16nm Products and Later [CSV File]
--ber_limit <limit_criteria>	BER Limit Criteria [Nominal(Default)/Corner/Drift] (Optional - Default Nominal)
--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 systems only.
--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 byte)
--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, separated by ',')

--margin	Read the SerDes eye margins per lane
--measure_time <time>	Measure time in seconds for single eye [10, 30, 60, 90, 120, 240, 480, 600 and 900] (Optional - Default 60 for PCIe and 30 for Network ports)
--eye_select <eye_sel>	Eye selection for PAM4 [UP, MID, DOWN, ALL] (Default ALL)
--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 bins. Each bin is holding different number of errored bit within FEC protected block
--show_histogram	Show FEC errors histogram
--clear_histogram	Clears FEC errors histograms
--yes	Non-interactive mode, answer yes to all questions

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
```

⚠ RX and TX lane rates for new devices includes 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]`

⚠ The PRBS pattern that is 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 info:

```
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 <polarity>,<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)
Device Status : N/A
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)
-----
: Pol,tap0,tap1,tap2,bias,preemp_mode,reg,leva
Lane 0 : 1,0,114,6,15,1,11,9
Lane 1 : 0,0,116,4,15,1,11,9

```

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 [EDR\25G or HDR\PAM4 speeds].

 If the margin scan fails with this message (Eye scan not completed), perform a reboot and run the scan again.

To enable the margin scan with measure time 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 (it 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 values of mixers:

```
mlxlink -d /dev/mst/mt4117_pciconf0 --rx_error_injection --mixer_offset0 0x200 --mixer_offset1 0x305
```

⚠ 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_error_injection --show_mixers_offset
```

Tool Usage with NIC vs. Switch (-p Flag)

When using mlxlink tool with NIC, notice that the "label_port" flag -p should not be used. To address different ports please use different mst devices.

For example:

To address port 1 in ConnectX-4 use:

```
mlxlink -d /dev/mst/mt4115_pciconf0
```

To address port 2 use:

```
mlxlink -d /dev/mst/mt4115_pciconf0.1
```

- ⚠**
- Any mlxlink command for switch should include the "-p" flag to address the specific port in the switch.
 - When working with the NIC, 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

Using mlxlink on NVIDIA Quantum HDR based switch systems split ports if the split port number is not provided by the ibdiagnet tool:

```
Mlxlink -d lid-<LID> -p <formula>
```

Formula:

In case of 2X port:

- 1- port_num = round_down[(lblinkinfo_port_num + 1)*0.5]
- 2- if (lblinkinfo_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
```

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]
Device Status                : [Device Status]
```

PCIe Switch

For NVIDIA® ConnectX®-5 and newer devices, the PCIe interface can be configured for PCIe switch. When the PCIe switch is enabled, the Depth, pcie_index and node parameters are needed to specify the PCIe port where the requested information (such as counters or eye info) is gathered from.

Parameters	Description
Depth	<p>This defines the number of layers from the Root Complex to the specific port.</p> <ul style="list-style-type: none">• For NVIDIA® ConnectX adapter cards multi-host mode, the Depth should be set to 0.• For NVIDIA® BlueField/BlueField-2 JBoF, the Depth should be set to 3.
Pcie_index	<p>This defines the root complex ID or host index.</p> <ul style="list-style-type: none">• 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	<p>This defines the specific pcie port.</p> <ul style="list-style-type: none">• For NVIDIA® ConnectX adapter cards multi-host mode, the node is always 0 for each host_index.• For NVIDIA® BlueField JBoF mode, this is from 0x0 - 0xF, accounting for up to 16 possible ports for BlueField JBoF.• For NVIDIA® BlueField-2, this is 0x0 - 0x7. <p>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</p>

Parameters	Description
	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)
Device Status                : N/A

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 following flag: `--port_type PCIE -c`

```
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 number of transitions to recovery due to Framing errors and CRC (dllp and tlp) errors.
- TX errors: indicate number of transitions to recovery 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)
Device Status                : Correctable Error detected, Unsupported Request detected.

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 following flag: `--port_type PCIe -e`

```
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)
Device Status                 : Correctable Error detected, Unsupported Request detected.

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

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.

⚠ mstresourcedump can be used only if Python 3.x is installed. Using lower versions will result in tool's failure.

⚠ It is important for the user to generate a bin file for debugging and troubleshooting cases when needed by NVIDIA.

⚠ 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] [--virtual-hca-id VIRTUAL_HCA_ID] --device DEVICE
```

where

-h, --help	Show help message and exit
--virtual-hca-id	The virtual HCA (host channel adapter, NIC) ID
-d, --device	The device name

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

```
resourcedump dump [-h] --device DEVICE --segment SEGMENT [--virtual-hca-id VIRTUAL_HCA_ID] [--index1 INDEX1] [--index2 INDEX2] [--num-of-obj1 NUM_OF_OBJ1] [--num-of-obj2 NUM_OF_OBJ2] [--depth DEPTH] [--bin BIN]
```

where

-h, --help	Show help message and exit
--virtual-hca-id	The virtual HCA (host channel adapter, NIC) ID
--index1	The first context index to dump (if supported for this segment)
--index2	The second context index to dump (if supported for this segment)
--num-of-obj1	The number of objects to be dumped (if supported for this segment). accepts: ["all", "active", number, depends on the capabilities]
--num-of-obj2	The number of objects to be dumped (if supported for this segment). accepts: ["all", "active", number, depends on the capabilities]
--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
--bin	The output to a binary file that replaces the default print in hexadecimal, a readable format

-d, --device	The device name
--segment	The segment to dump

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: 0xffff
Segment Size: 16 Bytes
Segment Data:
0x0004FFFE 0x00000000 0x00000000 0x101A0111
-----
Segment Type: 0xffffa
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.

 The tool applicable inputs for parsing can be the resourcedump outputs (bin file or the “human readable” format”), or the devlink json format output.

 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 --dump-file DUMP_FILE --adb-file ADB_FILE [-h] [--version] [--out OUT] [--raw] [-v]
```

where

--dump-file	Location of the dump file used for parsing
--adb-file	Location of the ADB file
-h, --help	Shows this help message and exit
--version	Shows the tool's version and exit

--out	Location of the output file
--raw	Prints the raw data in addition to the parsed data
-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 (0xffff9)
segment_header.segment_type = 0xffff9
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 (0xffffb)
segment_header.segment_type = 0xffffb
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 (0xffffa)
segment_header.segment_type = 0xffffa
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 (0xffff9)
segment_header.segment_type = 0xffff9
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 (0xffffb)
segment_header.segment_type = 0xffffb
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.

⚠ The pcap packet format is not supported by default, and requires installing MFT with the --with-pcap option.

⚠ 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

MFTt 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 mlx cables.

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 mlx cables example, refer to section [mlx cables](#).

mlx cables - Cables Tool

The mlx cables 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)

mlxables 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:

```
mlxables -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:

```
mlx cables -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:

```
# mlx cables -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:

```
# mlx cables -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:

```
mlx cables -d mt4115_pciconf0_cable_0 -g
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:

```
# mlx cables -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.

```

mlxables -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).

```

mlxables -d mt4115_pciconf0_cable_0 --read_reg vendor_oui
vendor_oui = 0x0002c9
mlxables -d mt4115_pciconf0_cable_0 --read_reg identifier
identifier = 0x0d

```

To read all the Eeprom of the cable:

```

mlxables -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:

```

# mlxables -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 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 mlx cables 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:

```

mlx cables -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:

```

mlx cables -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

```

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 (dev/mst/mt4113_pciconfx) in the output of mst status	For 5th generation (Group II) devices, there is only one method available for accessing the hardware. For example, Connect-IB device is represented by dev/mst/mt4113_pciconfx mst device	When querying a 5th generation (Group II) device, use the conf mst device (for example: dev/mst/mt4113_pciconfx)
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: <ul style="list-style-type: none">• 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: <ol style="list-style-type: none">1. Disable SRIOV in bios2. Reboot server3. Change num of VFs

Issue	Cause	Solution
		4. Enable SRIOV in bios
When Querying for current configuration on ConnectX-3/ ConnectX-3Pro, some of the parameters are shown as “N/A”	The current firmware on the device does not support showing the device's default configuration	Update to the latest firmware
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	<ol style="list-style-type: none"> 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 <p>Note: an additional reboot will be required after completing the installation</p>
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
<p>The following message is printed on screen when performing firmware update:</p> <p>An update is needed for the flash layout.</p> <p>The operation is not failsafe and terminating the process is not allowed.</p>	A flash alignment operation is required.	Approve the alignment, avoid process interrupt.

Issue	Cause	Solution
<p>Firmware update fails with the following message:</p> <p>-E- Burning FS4 image failed: Bad parameter</p> <p>Note: This is a rare scenario.</p>	Firmware compatibility issue.	Re-run the burn command with --no_fw_ctrl flag.
<p>The following message is printed on screen when performing firmware update:</p> <p>Shifting between different image partition sizes requires current image to be re-programmed on the flash.</p> <p>Once the operation is done, reload FW and run the command again</p> <p>Note: This is a rare scenario.</p>	Firmware compatibility issue.	Re-load firmware and re-run the burn command.
<p>The following message is printed on screen when trying to query/burn a Connect-IB device:</p> <p>-E- Cannot open Device: /dev/mst/mt4113_pciconf0. B14 Operation not permitted MFE_CMDIF_GO_BIT_BUSY</p>	Using an outdated firmware version with the Connect-IB adapter.	<ol style="list-style-type: none"> 1. Unload MLNX_OFED driver: /etc/init.d/openibd stop. 2. Add “-ocr” option to the 'flint' command. <p>For example: flint -d /dev/mst/mt4113_pciconf0 -ocr q</p>
<p>The following message is reported on screen when trying to remove the expansion ROM using the 'drom' option:</p> <p>-E- Remove ROM failed: The device FW contains common FW/ROM Product Version - The ROM cannot be removed separately.B9</p>	Updating only the EXP_ROM (FlexBoot) for recent firmware images which requires adding the 'allow_rom_change' option.	<p>Allow “-allow_rom_change” option to the “flint” command.</p> <p>For example: flint -d <mst_device> -allow_rom_change drom</p>
<p>Burning command fails and the following message is printed on screen:</p> <p>-E- Can not open 06:00:0: Can not obtain Flash semaphore (63). You can run "flint -clear_semaphore</p> <p>- d <device>" to force semaphore unlock. See help for details.</p>	<p>Semaphore can be locked for any of the following reasons:</p> <ul style="list-style-type: none"> • 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 currently burning the firmware 	<p>If no other process is taking place at the same time run the following command: flint -d <device> -- clear_semaphore</p> <p>OR</p> <p>Reboot the machine.</p>
<p>Burning tool fails with the following message:</p> <p>-E- Unsupported binary version (2.0) please update to latest MFT package.</p>	The binary version is incompatible with the burning tool.	Update MFT to the latest package.

Issue	Cause	Solution
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 mismatch: 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.

Issue	Cause	Solution
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: <ul style="list-style-type: none"> -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 supported.	N/A
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: <ul style="list-style-type: none"> 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.	Refer to the section Create Tokens for Secure Firmware and NV LifeCycle to learn how to generate and sign tokens.
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](#)

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.



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 than 16 characters, the remaining characters are filled with binary 0s by the burning tool. PSID Field Structure

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 Mellanox's MXML-CF128-T HCA board is MT_0030000001, where:

MT_	Mellanox vendor symbol
003	MXML-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:

⚠ 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:

- a. 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
```

- b. 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]>
```

- c. 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
```

⚠ 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.

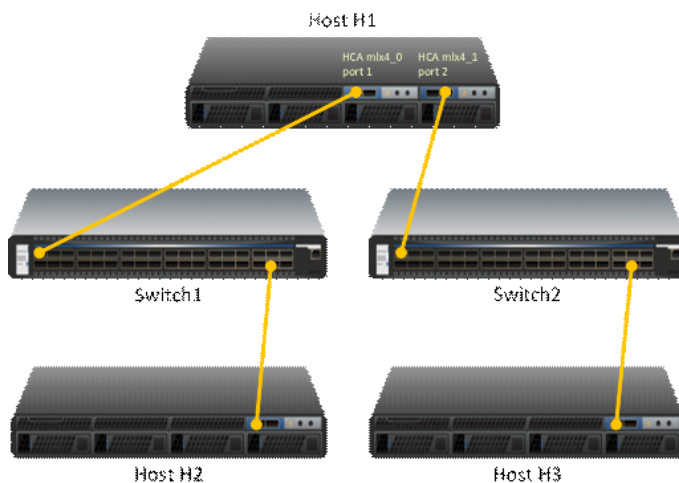
The device name would be in the format:

```
<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
```



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)

Item	Quantity	Description
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 *mst1* driver. Enter: (Note: This step is 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.
mst remote add <host- name>[:port]	<ul style="list-style-type: none">• Establishes connection with a specified host on a specified port (default port is 23108).

Command	Description
	<ul style="list-style-type: none"> 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
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 in order to re-burn the card, assuming that this is 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

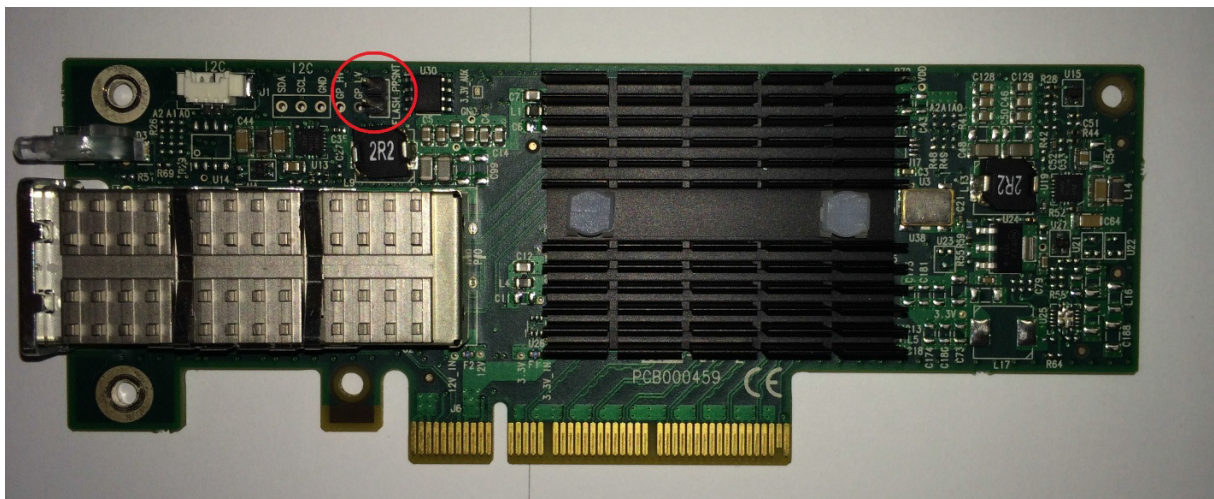
To boot the card in normal mode:

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.

⚠ 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.

 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 $((\text{guid} \gg 16) \& 0\text{xfffff}000000) \mid (\text{guid} \& 0\text{xfffff})$). 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. 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/mt511_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
```

a. To view flash settings, run:

```
# flint -d /dev/mst/mt511_pciconf0 -ocr hw query
```

b. To view assigned GUIDs, run:

```
# flint -d /dev/mst/mt511_pciconf0 -ocr q
```


c. 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.

 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
```



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
```

a. To view flash settings:

```
# flint -d /dev/mst/mt521_pciconf0 -ocr hw query
```

- b. To view assigned GUIDs:

```
# flint -d /dev/mst/mt521_pciconf0 -ocr
```

- c. To change a GUID after the initial burn:

```
# flint -d /dev/mst/mt4115_pciconf0 -ocr -guid 0xe41d2d0300570fc0 sg
```

- d. To change a MAC after the initial burn:

```
# flint -d /dev/mst/mt4115_pciconf0 -ocr -mac 0x0000e41d2d570fc0 sg
```

- e. 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.

 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-IB2 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 -wrmage 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-IB2 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.WriteProtected=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
```

- a. To view flash settings:

```
# flint -d /dev/mst/mt53000_pciconf0 -ocr hw query
```

- b. To view assigned GUIDs:

```
# flint -d /dev/mst/mt53000_pciconf0 -ocr q
```

- c. To change a GUID after the initial burn:

```
# flint -d /dev/mst/mt53000_pciconf0 -ocr -guid 0xe41d2d0300570fc0 sg
```

- d. To change a MAC after the initial burn:

```
# flint -d /dev/mst/mt53000_pciconf0 -ocr -mac 0x0000e41d2d570fc0 sg
```


- e. 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.

 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 -wimage 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-ected=Disabled  
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set Flash1.WritePro-ected=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
```

- a. To view flash settings:

```
# flint -d /dev/mst/mt583_pciconf0 -ocr hw query
```

- b. To view assigned GUIDs:

```
# flint -d /dev/mst/mt583_pciconf0 -ocr q
```

- c. 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.

As demonstrated in Section 2.3.11, you have to:

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, customer token and MLNX ID token. The XML must also have the file_applicable_to and file_mac_addr_list configurations.
2. Generate a binary file that can be applied to the device using the mlxconfig create_conf command (see Section 2.3.14).

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:

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
```

- 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
...
```

- 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
```

- 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 GuideNumber Base GUID: ec0d9a030048af2a 4 Base MAC: ec0d9a48af2a 4 Image VSD:
N/A Device VSD: N/A PSID: MT_0000000080 Security Attributes: N/A
```

- For the firmware update to take effect, you need to either reboot the server or run:

```
# mlxfwreset -d /dev/mst/mt4119_pciconf0 -y r
```

- 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 GuideNumber
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
```

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: <ul style="list-style-type: none">• ConnectX-3• ConnectX-3 Pro• SwitchX• SwitchX-2
5th Generation ICs/ Group II of ICs	Contains the following devices and newer: <ul style="list-style-type: none">• Connect-IB• Switch-IB• Switch-IB 2• Spectrum• ConnectX-4• ConnectX-4 Lx• ConnectX-5• ConnectX-5 Ex

Reference Documents and Downloads

Reference Documents and Downloads	Location
MFT web page	http://www.mellanox.com > Products > Firmware Tools > MFT - Firmware Tools.
Firmware images and their Release Notes	http://www.mellanox.com/page/firmware_download

Reference Documents and Downloads	Location
MLNX_OFED	http://www.mellanox.com > Products > Software > InfiniBand/VPI Drivers > Linux Driver.
WinOF/WinOF-2	http://www.mellanox.com > Products > Software > InfiniBand/VPI Drivers > Windows Driver.
VMware	http://www.mellanox.com > Products > Software > InfiniBand/VPI Drivers > VMware Driver.
FreeBSD	http://www.mellanox.com > Products > Software > InfiniBand/VPI Drivers > FreeBSD Driver.

User Manual Revision History

Date	Revision	Description of Changes
December 05, 2021	4.18.0	<ul style="list-style-type: none"> Updated section mlxlink Utility: <ul style="list-style-type: none"> Added "--tx_prbs <tx_prbs_mode>" & "--rx_prbs <rx_prbs_mode>" flags Added "--amber_collect" flag
June 30, 2021	4.17.0	<ul style="list-style-type: none"> Updated section Querying the Firmware Image, added Secure-boot attributes. Updated section mlxprivhost. Updated section Burning a Firmware Image, added the "--activate_delay_sec <timeout in seconds>" flag. Updated section mlxlink, added "--fom_measurement <eye>" flag.
March 31, 2021	4.16.3	<ul style="list-style-type: none"> Updated section mlxfwreset Updated section mlxlink: added FEC errors' histogram
March 01, 2021	4.16.2	<ul style="list-style-type: none"> Added section RX Error Injection Added section Hardware Security Module (HSM) Added section Secure Boot Updated section mlxconfig create conf Command Updated section flint - Firmware Burning Tool: added the --openssl_engine <engine name>, --openssl_key_id <key> and rsa_sign flags Updated section mlxconfig: added -eng --openssl_engine and -k --open_ssl_key_id flags
February 08, 2021	4.16.1	<ul style="list-style-type: none"> No changes to the User Manual.
January 04, 2020	4.16.0	<ul style="list-style-type: none"> Added section stedump Utility Added section Cable Firmware Update (In-Field-Firmware-Update) Updated section mlxconfig: added get_raw flag Updated section mlxprivhost: added the q,query flags Updated section mlxlink: added the --invert_tx_polarity & --invert_rx_polarity flags Updated section Margin Scan Tool
September 15, 2020	4.15.1	<ul style="list-style-type: none"> Updated section mlxlink: added Margin Scan for PCIe Link. Updated mlxmdio Utility Updated flint - Firmware Burning Tool
July 31, 2020	4.15.0	<ul style="list-style-type: none"> Updated the following sections: <ul style="list-style-type: none"> mlxlink mlxfwreset mlxcables: Removed the following flags: -u --update, -i --image-file <FileName>, f --force, --yes, --no
May 31, 2020	4.14.4	<ul style="list-style-type: none"> No changes to the User Manual.
April 30, 2020	4.14.2	<ul style="list-style-type: none"> Added section resourceparse Utility Updated section mlxlink Utility
February 28, 2020	4.14.0	<ul style="list-style-type: none"> Added section Burning the MFA2 Images

Date	Revision	Description of Changes
		<ul style="list-style-type: none"> Updated section MFT Supported Configurations and Parameters, added BOOT_INTERRUPT_DIS
December 12, 2019	4.13.3	<ul style="list-style-type: none"> Added the following sections: <ul style="list-style-type: none"> mlxfwstress Utility. resourcedump Utility Added a new registry key to mlxreg Utility Added the following section: <ul style="list-style-type: none"> Comparing the Binary Image
September 26, 2019	4.13.0	<ul style="list-style-type: none"> Added the "DYNAMIC_VF_MSIX_TABLE" parameter, see MFT Supported Configurations and Parameters Updated the following sections: <ul style="list-style-type: none"> mlxlink Utility. Added a new subsection Tool Usage on Quantum HDR Switch Systems with Split Ports mlxfwreset - Loading Firmware on 5th Generation Devices Tool
April 30, 2019	4.12.0	<ul style="list-style-type: none"> Updated the following sections: <ul style="list-style-type: none"> mlxfwmanager - Firmware Update and Query Tool: Added the following options: <ul style="list-style-type: none"> "--download-type Type" and "--ssl-certificate Certificate" flint - Firmware Burning Tool: Added the following options: <ul style="list-style-type: none"> "-qq", "--low_cpu" and "--flashed_version" mlxburn - Firmware Image Generator and Burner: Added the following options: <ul style="list-style-type: none"> "[-gb_bin_file <gb_bin_file>]" "-striped_image" "-vsd <string>" Updated the "--dev_type <mellanox-device-number>" option description Updated the "Additional mlxburn Options" mlxlink Utility: Added the following options: <ul style="list-style-type: none"> "--depth <depth>", "--pcie_index <pcie_index>", and "--node <node>" fwtrace Utility: Added the following options: <ul style="list-style-type: none"> "--gvmi" and "--ignore_old_events" Using mlxconfig to Split a Port in a Remotely Managed Switch
March 2019	4.11.0	Converted to online html format; some reorganization.

Release Notes Revision History

MFT Release Notes Change Log History

Component / Tool	Description	Operating System
Rev. 4.17.0		
Anti-rollback Protection	Enabled Anti-rollback protection to prevent old vulnerable firmware versions from being flashed to the device.	All
DSFP Modules	Added support for DSFP modules in mlxlink.	All
ESXi, VMware Certification	Downloadable ESXi files in MFT v4.17.0 are now certified by VMware.	ESXi
Remote mst Device Cable Support	Remote mst device now supports cable devices. The remote cables will be shown on the mst status and can be accessed via the mlx cables tool.	All
Parallel Firmware Burning in (DMA Burning)	<p>Added support for parallel firmware burning. Although DMA burning is supported in Virtual Machines as well, burning in such scenarios might be slower than on Physical Machines.</p> <p>Note: If the NIC driver is unloaded, burning via DMA is unsupported (due to BME is unset) and regular burn flow will be executed instead.</p> <p>Note: This capability is supported in 5th Generation devices only.</p>	Linux / FreeB SD
mlx_fpga	The mlx_fpga utility will be deprecated as of MFT v4.18.0.	All
Bug Fixes	See Bug Fixes .	All
Rev. 4.16.3		
mlxlink	<p>Added support for Rs FEC Histogram Counters in mlxlink. The result is divided to bins. Each bin holds a different number of errored bit within the FEC protected block.</p> <p>For further information, see mlxlink.</p>	All
MKey	<p>[Beta] Added support for Mkey. The MKEY field is used to authenticate SMP communication.</p> <p>Note: Mkey feature will work only with LID device.</p>	Linux
Bug Fixes	See Bug Fixes .	All
Rev. 4.16.1		

Bug Fixes	See Bug Fixes .	All
Rev. 4.16.0		
Cable Firmware Burning	<p>[Beta] Added support for LinkX module burning via MFT toolset. The new capability enables direct firmware burning from the internal flash storage to reduce the bandwidth and accelerate the burning process, including burning several modules at a time.</p> <p>For further information, see Cable Firmware Update (In-Field-Firmware-Update).</p>	All
MST Status	<p>The “mst status -v” command will now report RDMA bond devices mapped correctly to the corresponding ETH bond devices.</p> <p>Note: Does not support RDMA Bonding for Socket Direct.</p>	Linux
mlxconfig	Added the following new configuration option in mlxconfig to control the Physical link parameter on boot: DEFAULT, LEGACY and ADVANCED.	All
stedump Utility	<p>The stedump tool is a packet simulator for host NIC steering solutions. The dump output of hardware steering is used for debugging and troubleshooting.</p> <p>For further information, see stedump Utility.</p>	Linux
mlxlink	<p>Enabled margin scan on Network links.</p> <p>For further information see mlxlink.</p>	All
mlxlink	<p>Added PRBS TX/RX polarity inversion using the following flags: --invert_tx_polarity / --invert_rx_polarity</p> <p>For further information see mlxlink.</p>	All
mlxprivhost	<p>Enabled querying the current host configuration using the “q query” flag.</p> <p>For further information see mlxprivhost.</p>	Linux
mlxconfig	<p>Now the user can get raw configuration using “get_raw” flag.</p> <p>For further information see mlxconfig.</p>	All
General	See Bug Fixes	All
Rev. 4.15.1		
mlxlink	<p>Added support for PCIe eye grade scan.</p> <p>Note: This feature is at beta level for the network ports.</p> <p>Note: When using a Multi-host and a Socket Direct system, you must specify the the port or the DPN (depth, pcie_index, node). The links can be shown using the “--show_links” flag on the PCIe port.</p> <p>For further information see mlxlink and Margin Scan for PCIe Link.</p>	All
General	See Bug Fixes	All
Rev. 4.15.0		

Adapter Cards	Added support for NVIDIA® ConnectX®-6 Lx adapter card.	All
Adapter Cards	Added support for NVIDIA® BlueField-2 SmartNIC adapter card.	All
mlxfwreset	<p>Enabled the driver and the firmware to synchronize the reset between all hosts using the mlxfwreset utility. This new capability can be run from one of the hosts instead of all of them.</p> <p>This capability can be activated by setting the new flag "--sync" to 1.</p> <p>Note: The new mlxfwreset sync capability (--sync) is available only if supported by the firmware and all the drivers on all the hosts. To check if this is supported, run the "query" command.</p> <p>For further information see mlxfwreset.</p>	Linux, Multi-Host
mlxfwreset	Enabled running mlxfwreset from both the host and Arm while the NVIDIA® BlueField Smart-NIC is in isolated mode.	All
mlxfwreset	<p>Added a new error message when trying to run mlxfwreset on Windows OS and the PowerShell.exe is not installed on the machine.</p> <p>The error message is: "-E- PowerShell.exe is not installed. Please stop the driver manually and re-run the tool with --skip_driver."</p>	Windows
mlxlink	<p>Enabled PRBS test mode for Multi-Host and host-management devices.</p> <p>Note: For Multi-Host devices, another interface should be maintained to enable the link back.</p>	All
mlxlink	<p>Enabled working with ports group mapping for NVIDIA® Spectrum-2 and NVIDIA® Quantum switches.</p> <p>For further information see mlxlink.</p>	All
mlxlink	Added support for NVIDIA® Spectrum-3 based switch systems.	All
mlxlink	<p>Added support for QSFP-DD and CMIS cables for mlxlink.</p> <p>For further information see mlxlink</p>	
mlxreg	Added new access registry keys.	All
General	See Bug Fixes	All
Rev. 4.14.4		
MTCR	Added MTCR Python API to WinMFT package.	Windows
Rev. 4.14.2		
General	Added support for arm64 architecture to Windows OS.	Windows
mlxlink	Extended reading and writing the Serdes Transmit Parameters for ConnectX-6 and ConnectX-6 Dx adapter cards.	All

mlxlink	Added support to access the module information including reading the Digital Diagnostic info, dump EEPROM pages, read/write to specific module page. For further information, see the new cable flags and cable operations in mlxlink Utility .	All
mlxlink	Added support for all available PRBS patterns for each device like (Square wave patterns and PRBS13 patterns).	All
mlxlink	Added configuration for PRBS test mode per lane. For further information, see the "-lanes" flag in mlxlink Utility .	All
resourceparse	The resourceparse tool parses and prints data segments content. The parser's output is used by NVIDIA® representatives for debugging and troubleshooting. For further information, see resourceparse Utility .	Linux / Windows
Rev. 4.14.1		
Arm architecture	[Beta] Added support for Arm64 architecture to ConnectX-4 Lx adapter cards.	Windows
Rev. 4.14.0-105		
resourcedump	Added support for "--virtual-hca-id" command. Now the tool can provide info on the virtual HCA (host channel adapter, NIC) ID. For further information see resourcedump Utility	Linux / Windows
mlxlink	HDR lane rate is now supported when in Pseudorandom Binary Sequence (PRBS) mode.	All
mlxreg	Increased the registry keys the tool supports and now it exposes the full PRM. For additional information, refer to the PRM.	All
mlxconfig	BOOT_INTERRUPT_DIS parameter was added to mlxconfig. When TRUE, legacy interrupts should not be used for receive/transmit indication. Polling should be used instead. Note: This is supported only if boot_legacy_interrupt_disable_supported is set to TRUE.	All
mlxlink	mlxlink output can be printed now in JSON format by using the "--json" flag.	All
flint	Enables the user to to insert information manually to the flash on components such as MFG/DEV GUID/MAC when no information exists after the burn process using the command "flint -d <device> sg <guid>". If the information is not inserted manually, the existing GUID/MAC information will be used instead.	All
mlxlink	Added supported for switching between NRZ/PAM4 speeds for new devices that support HDR/200G speeds (ConnectX-6, ConnectX-6 Dx, NVIDIA® Quantum, NVIDIA® Spectrum 2).	All
Rev. 4.13.3		
Binary Image Comparison	Enables the user to verify a firmware image on a device which operates in livfish mode by comparing it with an existing binary firmware file. For further information see Comparing the Binary Image .	Linux / FreeBSD

SDK	Added two new libraries to the WinMFT package for developing software that interacts with NVIDIA® devices The new SDK includes the mtrc and fastfwreset libs and headers.	Windows
resourcedump	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. For further information see resourcedump Utility . Note: This utility is supported only on Python 3.0 and up.	Linux Windows
mlxreg	Added a new registry key: NCFG. This register is used to enable/disable device features and it is supported when ICMD_QUERY_CAPABILITY.ncfg_reg==1. For further information see mlxreg Utility	All
Rev. 4.13.0		
Dynamic MSI-X Allocation	Dynamic MSI-X allocation capability allows users to control the number of MSI-X vectors allocated to a Virtual Function, thus, improve performance in guests systems. For further information of how to set this capability, see the "DYNAMIC_VF_MSIX_TABLE" parameter, in section MFT Supported Configurations and Parameters .	Windows
Fast Firmware Reset	Added support for a fast firmware reset (< 1 second) to ConnectX-5 adapter cards.	Windows
mlxfwreset	[Beta] Added support for Socket Direct devices on Windows. Note: Please be aware, due to its quality level support, occasionally, bluescreens might occur.	Windows
mlxlink	Added support for reading the "Link Downed Counter" and "Link Error Recovery Counter" in the mlxlink utility when using InfiniBand protocol only.	All
mlxlink	Added support for HDR PCIe grades in the EYE Opening Info in the mlxlink utility.	All
mlxlink	Added a new flag (show links) to define the valid PCIe links. For further information, refer to mlxlink Utility examples.	All
mlxconfig	Added the ATS_ENABLED TLV param. When set to TRUE, the device will support Address Translation Service (ATS).	All
mlxfwreset	Added save/restore ATS PCIe capability.	All
mlxarchive	Added support for MFA2 query using the mlxarchive tool. For further information refer to mlxarchive - Binary Files Compression Tool .	Linux FreeBSD
mlxfwreset	Added support for Live-Patch in ConnectX-5.	All
Mitigation Techniques	Added HIGHENTROPYVA and LARGEADDRESSAWARE mitigation techniques. <ul style="list-style-type: none"> HIGHENTROPYVA - high-entropy 64-bit address space layout randomization (ASLR) LARGEADDRESSAWARE - indicates that the application can handle addresses larger than 2 gigabytes 	All

Preboot Boot Settings	Updated the LEGACY_BOOT_PROTOCOL settings, added an NVME option. For further information refer to MFT Supported Configurations and Parameters .	All
mlxfwreset	[Beta] Added a new reset option (reset-type) to the reset command of mlxfwreset. The user can see the supported reset-types by using the query command. For further information refer to mlxfwreset - Loading Firmware on 5th Generation Devices Tool .	All
mlxconfig	Added the VF_VPD_ENABLE parameter to mlxconfig. When set, the VPD capability is exposed to Virtual Functions.	All
OpenSSL	Updated the OpenSSL to 1.0.2s.	All
mst Status	Updated the way the GUID is displayed when running "mst status" on unmanaged switch systems. For example, <ul style="list-style-type: none">• Before the change: <code>/dev/mst/SW_MT53000_7cfe900300c09830_lid-0x0036 /dev/mst/SW_MT54000_98039b0300867b9a_lid-0x0012</code>• After the change: <code>/dev/mst/ SW_MT53000_SwitchIB_Mellanox_Technologies_lid-0x0036 /dev/mst/ SW_MT54000_Quantum_Mellanox_Technologies_lid-0x0012</code>	All
Rev. 4.12.0		
.deb Package Name	Changed the name of *.deb files from " <i>mft-<version>.amd64.deb</i> " to " <i>mft_<version>_amd64.deb</i> " e.g., from <i>mft-4.11.0-34.amd64.deb</i> to <i>mft_4.11.0-34_amd64.deb</i>	Linux
General	Added support for Spectrum-2 based switch systems.	All
Cables	Added support for HDR cables in mlx cables and mlxlink.	All
mlxfwmanager	Enabled the option to query PLDM images in mlxfwmanager.	All
mlxlink	mlxlink adjustment to enable an easier read of the access register MPEIN due to its structure change. MPEIN access register now works according to depth and pcie_index, node.	All
fwtrace	Added support for fwtrace in secure firmware without cs_token.	Linux (kernel 4.19 and above)
Switch Firmware	Enabled the option to extract the firmware ISSU version from the switches' firmware image.	Linux / MLNX-OS
Zero Touch RoCE	Added support for Zero Touch RoCE. It enables RoCE to operate on fabrics where no PFC nor ECN are configured. This makes RoCE configuration a breeze while still maintaining its superior high performance.	All
flint	Enabled setting VSD when Memory Chip Controller (MCC) capability is enabled.	All

flint	Added an option to reduce CPU utilization with "--low_cpu" flag.	All
General	Removed the CONFIG COMPACT definition.	Linux
General	Added support for libibmad 12.	Linux
mlxconfig	Renamed the BOOT_RETRY_CNT1 parameter to BOOT_RETRY_CNT.	All
Bug Fixes	See MFT Bug Fixes History	All
Rev. 4.11.0		
.deb Package Name	As of MFT v4.12.0, the name of *.deb files will be changed from "mft-<version>.amd64.deb" to "mft_<version>_amd64.deb" e.g., from mft-4.10.0-104.amd64.deb to mft_4.10.0-104_amd64.deb	All
Supported Devices	Added support for NVIDIA® Quantum switch systems and ConnectX-6 Ready adapter cards. For further information on the ConnectX-6 adapter cards, please contact Support .	All
mlxarchive tool	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. For further information, refer to section mlxarchive .	Linux FreeB SD
mlxprivhost	The ability to restrict the hosts from configuring the NIC. Meaning, only the Arm side will have the privilege to configure the NIC. Note: This utility is supported in BlueField devices only.	All
mlxconfig in BlueField	Enables the user to manage (grant/restrict) mlxconfig configuration privileges for BlueField Arm systems.	All
Bug Fixes	See MFT Bug Fixes History	All
Rev. 4.10.0		
ESXi	Added support for ESXi 6.7.	ESXi
FreeBSD	Added support for verbose output when running "mst status" in FreeBSD.	FreeB SD
mlxfwreset	Enabled mlxfwreset loading/unloading of the driver per a specific device in Linux OSes. Note: On Multi Host devices with firmware version lower than 1x.23.xxxx, the flag "--pci_link_downtime 2.5" must be added to mlxfwreset	Linux
Secure Firmware	flint now handles all the burn parameters when MCC is enabled and displays the secure-FW CS tokens.	All
Supported Devices	[Beta] Added support for BlueField™ SmartNIC.	All
Mlxconfig	Added the option to query partial parameters	All

Mlxconfig	Added the following new parameters: <ul style="list-style-type: none"> • FLEX_PARSER_PROFILE_ENABLE • ECPF_ESWITCH_MANAGER • ECPF_PAGE_SUPPLIER • SAFE_MODE_ENABLE • SAFE_MODE_THERSHOLD • BOOT_UNDI_NETWORK_WAIT 	All
Bug Fixes	See MFT Bug Fixes History	Linux
Rev. 4.9.1		
mlxfwreset	Added support for mlxfwreset in Power9 platforms.	Linux
Rev. 4.9.0		
mlxfwreset	Added support for a hot swap (or hot plug) of the PCIe slot.	Linux
Secure Firmware Update	Added support for Secure Firmware Update to ConnectX-4 adapter cards.	All
	Enabled signing the package with an RSA 4096 bit keys.	All
	Added support for setting the GUIDs when Secure Firmware Update is enabled.	All
mlxconfig	Added the following mlxconfig configuration parameters: <ul style="list-style-type: none"> • AUTO_RELOAD • DRIVER_SETTINGS • EXP_ROM_PXE_ENABLE • EXP_ROM_UEFI_ARM_ENABLE • EXP_ROM_UEFI_X86_ENABLE • INTERNAL_CPU_MODEL • IPV4 • IPV6 • PCI_DATA_WR_ORDERING_MODE • PXE_UNDI • STATUS_UPDATE • TCP • TCPIP • TRACER_ENABLE 	All
mlxlink	Added support for force speed configuration.	All
	Added support for the PEPC (show_external_phy) register.	All
mlxdump	Added support for nvlog dump.	All
Rev. 4.8.0		
mlxconfig	Added support for hardware timestamp in ConnectX-3/ConnectX- 3 Pro devices.	All
	Added the following mlxconfig configuration parameters: <ul style="list-style-type: none"> • MULTI_PORT_VHCA_EN • BOOT_LACP_DIS 	All

	<ul style="list-style-type: none"> • IP_OVER_VXLAN_PORT • IP_OVER_VXLAN_EN • UEFI_HII_EN • IB_ROUTING_MODE_P1 • IB_ROUTING_MODE_P2 • SRIOV_IB_ROUTING_MODE_P1 • SRIOV_IB_ROUTING_MODE_P2 	
Secure Firmware Update	Added support for Secure Firmware Update in ConnectX-5/ConnectX-5 Ex.	All
	Added support for setting forbidden versions.	All
FPGA management for JTAG Programming	Added the option to enable/disable FPGA management by the firmware for JTAG programming.	Linux
Rev. 4.7.0		
MST driver Microsoft certification	MST driver Microsoft certification allows running tools in extended secure boot environment.	Windows
Secure Firmware Update	Added support for Secure Firmware Update in ConnectX-4 and ConnectX-4 Lx.	Linux
flint	Added sign command for secured images. .	Linux
	Added a flag to enforce working in a non-secure mode, if available (according to security type).	
	Added expansion ROM CPU architecture to the flint query when the expansion ROM is available.	All
mlxlink	Added a new tool that displays and configures port related data at the physical layer.	All
mlxconfig	Added new mlxconfig TLVs.	All
	Added support for generating and applying TLV configuration files.	All
mlxdump	Added a new dump type “fsdump” to support dumping flow steering tables.	All
mst	Added support for adding remote devices in mst remote when the target machine does not have an MST kernel loaded.	Linux
mlx cables	Added the option to dump the data from all readable pages.	All
	Added support for burning cable firmware on In Service Firmware Update (ISFU) supporting cables.	All
	Added support to access the cable via the MTUSB, when the cable is connected to a compatible board.	All
mlxfwreset	Added support for MultiHost platforms.	All

Rev. 4.6.0		
Adapter Cards	Added support for ConnectX-5/ConnectX-5 Ex adapter cards. Note: ConnectX-5/ConnectX-5 Ex adapter cards are currently at Beta level.	All
mlxconfig	Added an option to query active (current) configurations in mlxconfig.	All
	Added new parameters in VPI settings configuration: XFI_MODE, PHY_TYPE, FORCE_MODE	
	Added a new parameter to the PCI configuration NON_PREFETCHABLE_PF_BAR	
mlxburn	Added the ability to use mlxvpd to read the device VPD when using mlxburn.	Linux, Windows, VMware ESXi
fwreset	Added support for fwreset in PPC64 and PPC64LE platforms.	Linux
Rev. 4.5.0		
General	Added support for Innova IPsec 4 Lx EN /Innova Flex 4 Lx EN	Linux
	MFT package size has been reduced in Linux by separating the architecture specific RPMs, and in ESXi, by moving relevant tools to the OEM package.	Linux / ESXi
mlx cables	Enhanced cable query capabilities. Added the additional registers below for debug purposes when running the query (-q) flag: <ul style="list-style-type: none"> • device technology • identifier • wavelength/attenuation • speed/compliance 	All
	Added a new query to read thresholds and monitor the cable's properties: <ul style="list-style-type: none"> • Temperature • Voltage • RX/TX powers • TX Bias 	All
	Added a new RAW format for printing the data of the cable's pages using the "--raw/--format raw" flags.	All
mlxconfig	Enabled mlxconfig to work with a database that describes the meta data of the TLVs configuration of fifth generation devices.	All
	Added the following configuration TLVs to mlxconfig: <ul style="list-style-type: none"> • MPFS • KEEP LINK UP • SW OFFLOAD CONF 	All

mlxreg	Added support for PPTT, PPRT and PPAOS access registers in switches.	All
flint	Added support for viewing and changing OEMs' device flash parameters using an IB device when using flint.	All
Rev. 4.4.0		
mlxfwreset	Added support for mlxfwreset in PowerPC	Linux
mlxconfig	Added the following new configurations: <ul style="list-style-type: none"> • Number of TCs • Number of VLs • Enable DCBX in CEE mode • Enable DCBX in IEEE mode • Allow the NIC to accept DCBX configuration from the remote peer • Enable DCBX • Enable the internal LLDP client • Select which LLDP TLV will be generated by the NIC 	All
General	Added support for all tools to work when the MST driver is not installed	Linux
mlx cables	Added support for dumping NVIDIA® cables EEPROM by mstdump/mlxdump tools	Linux Windows FreeBSD
	Added a new tool (mlx cables) that reads/writes NVIDIA® cable registers and queries the cables info	Linux Windows FreeBSD
Build	Created one MFT package for all 64 bits FreeBSD OSs	FreeBSD
mlxfwmanager_pci	Removed support for mlxfwmanager_pci tool (it is deprecated), since all the Linux tools can work without a kernel now. When required, use mlxfwmanager instead.	Linux
mcra	Added support for clearing VSEC PCI semaphore by the mcra tool. The new capability can be used after killing a tool forcefully without clearing the semaphores. Supported devices: ConnectX-4, ConnectX-4 Lx and Connect-IB	All
mlxreg	Added support for Switch-IB, Switch-IB 2 and Spectrum in the mlxreg tool	All
mlxconfig	Added the mlxconfig tool to the MFT package for WinPE	Windows
mlxconfig	Added a backup command in mlxconfig which allows user to save backup of the non-volatile configurations in a RAW file. This file can be set on the device by using the set_raw command	All

Build	Added support for running wrapped python tools (like fwtrace) in PPC64, PPC64LE and Arm platforms	Linux
mlxreg	Added support for PPRT and PPTT registers in ConnectX-4 and ConnectX-4 Lx	All
Rev. 4.3.0		
General	Added support for Spectrum device.	All
	Added support for Switch-IB 2 device.	All
	Added support for ConnectX-4 and ConnectX-4 Lx in VMware Esxi.	VMware ESXi
	Added support for VMware ESXi 5.5 Native.	VMware ESXi
	4th generation and 5th generation IC devices are now also named Group I ICs and Group II ICs, respectively.	N/A
mlxconfig	Added support for setting some of the parameters in textual values in addition to numerical values.	All
	Added new configurations: <ul style="list-style-type: none"> • The PF log bar size • The VF log bar size • The number of PF MSIX • The number of VF MSIX • port owner • Allow RD counters • IP protocol used by flexboot 	All
	Added the option to display the configuration's default values.	All
flint	Added support to calculate checksum on selected sections in the firmware image.	All
	Added the option to attach a timestamp to the firmware image.	All
Burning Tools	Improved firmware burn performance in livefish mode on 5th generation devices.	All
	Added the ability to show the running firmware version in case it does not match with the burnt firmware version on the flash. This case generally occurs after firmware upgrade and before firmware reload.	All
mlxreg	Added support for mlxreg tool which can be used to modify access registers or to query them.	All
mst	Created an mst device per physical function. It can be seen by running 'mst status -v'.	All

mlxfwmanager	Added support to create self-extractors in VMware ESXi OSs.	VMware ESXi
fwtrace	Added support for the fwtrace tool in FreeBSD.	FreeBSD
mlxfwreset	Added support for mlxfwreset in Windows and FreeBSD.	Windows FreeBSD
Rev. 4.1.0		
General	Added support for ConnectX®-4 Lx	Linux / Windows / FreeBSD
	Added support for ConnectX®-4	FreeBSD
mlxconfig	Added support for the following configurations in ConnectX-4, ConnectX-4 Lx and Connect-IB: <ul style="list-style-type: none"> • IB Dynamically Connect • Internal Settings • RoCE Congestion Control ECN • RoCE Congestion Control Parameters • Wake on LAN 	Linux / Windows / FreeBSD
	Added support for the following configurations in ConnectX-3 and ConnectX-3 Pro: <ul style="list-style-type: none"> • InfiniBand Boot Settings • Preboot Boot Settings 	Linux / Windows / FreeBSD
mlxtrace	Added support for MEM mode in ConnectX-4	Windows
cpld_update	Added the cpld_update tool to the OEM package	Linux
mlxfwreset	Added support for resetting the firmware	Windows / FreeBSD
fwtrace	Added support in FreeBSD	FreeBSD
Burning Tools	This version supports new ConnectX-4/Connect-IB firmware version format (<u>MM.mm.ssss</u>). It also enables upgrade of older firmware version format: MM.mmmm.ssss	All

Rev. 4.0.0		
General	Added support for ConnectX-4 device	Linux / Windows
	Removed support for ConnectX and ConnectX-2	All
mlx_fpga	Added a new tool that dumps registers and burns hardware for FPGA	Linux
mlxconfig	Added support for ConnectX-4 and Connect-IB (Beta level)	Linux / Windows / VMware ESXi
mlxfwmanager	Added support for FreeBSD and VMware ESXi	FreeBSD / VMware ESXi
mlxburn	Added support for VMware ESXi	VMware ESXi
Rev. 3.8.0		
General	Added support for Switch-IB device (at beta level)	Linux / Windows
	Added support for Debian/Ubuntu in PPC64 platform	Linux
	Added support for ESXi 2015 OS (Native)	VMware ESXi
mlxphyburn	Added support for burning Aquantia external PHY	Linux
mlxconfig	Added support for changing BAR size parameter	Linux / Windows / VMware ESXi
Rev. 3.7.1		

Bug Fixes	See MFT Bug Fixes History	Linux / Windows/ VMware ESXi/ FreeBSD
Rev. 3.7.0		
mlxfwmanager	Added online firmware update	Linux / Windows/ VMware ESXi
mlxburn	Added concurrency support to VPD read	Linux / Windows
	Added mlxburn to MFT	FreeBSD
flint	Added concurrency support to query firmware	Linux / Windows/ VMware ESXi/ FreeBSD
General	Added support for Arm platform and Power8	Linux
	Removed support for x86	Windows
mlxfwreset	Firmware reset for Connect-IB	Linux
fwtrace	Added fwtrace tool	Windows
Rev. 3.6.1		
mlxconfig	Added mlxconfig tool for changing non volatile configuration on device	Windows
Burning Tools	Added support for micron flash in flint and updated production burn flow on Connect-IB	Windows

Rev. 3.6.0		
mlxconfig	Added mlxconfig tool for changing non volatile configuration on device	Linux / VMware ESXi
Burning Tools	Added support for micron flash in flint and updated production burn flow on Connect-IB	Linux / VMware ESXi
mtserver	Added support for mstserver	FreeBSD
Rev. 3.5.1		
package content	Added support for the following tools: mst, mlxfwmanager, itrace, mlxtrace, mlxdump, mlxmcg, wqdump, mcra, mget_temp, pckt_drop, mlxuptime	VMware ESXi
flint mstdump	Added support for ConnectX®-3 Pro	VMware ESXi
	Redesigned the utility to make its look and feel more user friendly	VMware ESXi
	Added support for ConnectX®-3 Pro	VMware ESXi
Rev. 3.5.0		
flint/wqdump	Redesigned the flint and wqdump utility to make their look and feel more user friendly	Linux / Windows
flint	Added support for brom in Connect-IB®	Linux / Windows
mlxmdio	Added support for the mlxmdio utility	Linux
mlxfwmanager	Added support for Connect-IB	Linux / Windows

FreeBSD	Added support for FreeBSD operating system (at beta level)	FreeBSD
Rev. 3.1.0		
General	<p>The MFT package now has 2 installation flavours - standard (default mode) and 'OEM'. The OEM mode provides the following extra functionality:</p> <ul style="list-style-type: none"> Tools for creating mlxfwmanager package Several features for flint that are used in Connect-IB™ production 	Linux
Flint	Added support for burning Connect-IB™ via firmware interface. The '-override_cache_replacement' flag is not needed. This provides a 'safe' firmware update flow, without the risk of firmware or driver hanging	Linux
mlxfwmanager	Added support for the mlxfwmanager utility (at Beta level)	Linux
mlxuptime	Added support for the mlxuptime utility (at Beta level)	Linux
Rev. 3.0.0		
General	Added support for Connect-IB™ device (at beta level)	Linux / Windows
	Added support for ConnectX®-3 Pro device (at beta level)	Linux / Windows
	Added support for Ubuntu operating system	Linux
	Added support for running tools against PCI device [domain]:bus:dev.fn like: 0000:1a:00.0 or 1a:00.0 and devices used by OFED driver like: mlx4_0	Linux
	The package contains only the flint firmware update tool. Other debug tools were removed	Windows
flint	Added support for new flash types: N25Q0XX (Micron) and W25Xxx (Winbond)	Linux / Windows
mlxdump	Added support for the mlxdump utility (at beta level)	Linux / Windows
mlxmcg	Renamed mcg to mlxmcg	Linux / Windows

spark	spark was removed from MFT version 3.0.0	Linux / Windows
Supported Devices	<p>The following adapter cards and switch systems are no longer supported in MFT version 3.0.0:</p> <ul style="list-style-type: none"> • InfiniHost 4X • InfiniHost III Ex • InfiniHost III Lx 4X • InfiniScale • InfiniScale III 	Linux / Windows
Rev. 2.7.2b		
All	Added support for WinPE 4.0 OS	Windows
Rev. 2.7.2		
General	It is no longer required to run mst start/stop when using WinMFT tools. The service is automatically loaded/unloaded when an MFT tool is running. The mst service installation was removed from the setup	Windows
	Added support for SwitchX® silicon devices	Windows
flint	Added support for Atmel AT25DFxx flash family	Windows
	Added support for burning firmware via Command Line Interface (CLI) on SwitchX® devices	Windows
mget_temp	mget_temp displays a more accurate temperature reading for ConnectX®-2 and ConnectX®-3 devices by using the adapter's specific thermal calibration data	Windows
Rev. 2.7.1a		
Added the mcg tool (Beta level)	<p>The mcg tool displays the current multicast groups and flow steering rules configured in the device.</p> <p>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.</p> <p>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</p>	Linux
Removed support for In-band access on OFED 1.4 InfiniBand driver	In-band access is supported using OFED 1.5.X and higher	Linux
Rev. 2.7.1		

General	Added mlxconfig tool. This tool sets firmware configurations for NVIDIA® adapters. These configurations are nonvolatile they apply over device reboots. For further details, please run “mlxconfig -h”. The tool is at beta level	Linux
	Added support for NVIDIA® ConnectX®-3 silicon device	Windows
	Added the I2CBridge (Dimax's Driver for USB to I2C Adapter) as part of the WinMFT installation package. However, the I2CBridge is not installed by default	Windows
MFT installation change	Removed the isw tool. The isw tool functionality was replaced by the "mlx2c" tool. For example, to scan the devices on the i2c bus, run: > mlx2c -d <dev> scan instead of > isw -d <dev>	Windows
mget_temp	mget_temp displays a more accurate temperature for ConnectX-2 devices by using chip specific thermal calibration data	Linux
flint	Added support for Atmel AT25DFxx flash family	Linux
	Cleared error messages displayed when trying to burn firmware image of a different device. For example when burning ConnectX-2 firmware image on ConnectX-3 device	Linux
	Added support for flash type SST25VF016B	Windows
	Added support for flash type M25PX16	Windows
	<ul style="list-style-type: none"> The ROM section in the image now contains multiple boot images. Therefore flint was modified to display information for all of the images in the ROM section. Added support to display/burn UEFI ROM/ 	Windows
	Added an option to set the VSD and GUIDs in a binary image file. This is useful for production to prepare images for pre-assembly flash burning. These new commands are supported by NVIDIA® 4th generation devices	Windows
	Added an option to set the VSD and GUIDs on an already burnt device. These commands (“sg” and “sv”) re-burn the existing image with the given GUIDs or VSD. When the 'sg' command is applied on a device with blank (0xff) GUIDs, it updates the GUIDs without re-burning the image	Windows
mst	Added support for using ibnetdiscover in the 'mst ib add' command	Windows
mlxburn	Added support for VPD read/write	Windows
Rev. 2.7.0a		
Bug Fixes	See MFT Bug Fixes History	Linux
Rev. 2.7.0		

General	Added support for NVIDIA® ConnectX®-3 and SwitchX™ silicon devices	Linux
	Added Secure host feature which 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 (see flint changes). MFT tools cannot run on a device with hardware access disabled. This feature is enabled only with supporting firmware	Linux
	Removed support for Itanium (ia64)	Linux
flint	Added the following commands: <ul style="list-style-type: none"> • enable/disable access to the hardware • set/change the key used to enable access to the hardware 	Linux
	The ROM section in the image now contains multiple boot images. Therefore the flint was modified to display information for all of the images in the ROM section	Linux
	Added support to display/burn UEFI ROM	Linux
	Added support for burning firmware via Command Line interface on SwitchX devices	Linux
Mlxburn	Added option to add or replace a single keyword in the VPD writable section (-vpd_set_keyword flag)	Linux
	Added the option to set a binary VPD field data	Linux
MFT installation	Added the option --without-kernel which allows user to install MFT without the mst kernel	Linux
Rev. 2.6.2		
MFT installation change	RPM based installation: <ul style="list-style-type: none"> • Applications are installed using a pre-compiled binary RPM • Kernel modules are distributed as a source RPM and compiled by the installation script • Fast installation process 	Linux
	Removed prerequisite libraries: expat and zlib-devel	Linux
	The package tools, libraries and headers are now installed under:{ prefix }/bin or { prefix }/lib and { prefix }/include dirs. Directory /usr/mst is not created. For example, the “mread”, “mwrite” and “mcra” tools that were previously installed by default under /usr/mst/bin, now are installed under /usr/bin	Linux
		Linux
	Removed the InfiniScale® and InfiniBridge® tools	Linux
	Removed the Infinivision tool set	Linux

	Removed the isw tool. The isw tool functionality was replaced by the "mlx2c" tool. For example, to scan the devices on the i2c bus, run: > mlx2c -d <dev> scan instead of > isw -d <dev>	Linux
flint	Added support for flash type SST25VF016B	Linux
	Added support for flash type M25PX16	Linux
	Added an option to set the VSD and GUIDs in a binary image file. This is useful for production to prepare images for pre-assembly flash burning. These new commands are supported by NVIDIA® 4th generation devices	Linux
	Added an option to set the VSD and GUIDs on an already burnt device. These commands ("sg" and "sv") re-burn the existing image with the given GUIDs or VSD. When the 'sg' command is applied on a device with blank (0xff) GUIDs, it updates the GUIDs without re-burning the image	Linux
mst	Added support for using ibutils2/ibdiagnet and ibnetdiscover in the 'mst ib add' command	Linux
	Removed the _uar, _msix and _ddr devices from the mst device list	Linux
Debug tools	Added support for routing I2C bus to the IS4 device on IS50XX systems	Linux
Rev. 2.6.1		
Bug Fixes	See MFT Bug Fixes History	Linux
Rev. 2.6.0		
MFT installation change	Added the options: --without-image-generation, --disable-dc, and --without-kernel which allow for a partial installation in order to avoid problems with SW dependencies	Linux
	Now allows a non-root user to prepare MFT RPMs	Linux
All	Added NVIDIA® ConnectX®-2 and BridgeX® support	Linux / Windows
flint	Added a CRC check for the full image	Linux
	Support for query/burn of clp-gpxe ROM	Linux
	Prevents burning a ConnectX-2 image onto a ConnectX device and vice versa	Linux
	Added a logging option to flint	Linux

	<p>For the ConnectX device family only:</p> <p>Added commands for an independent burn/read/remove of an Expansion ROM image.</p> <p><i>For firmware versions earlier than 2.7.000:</i> It is possible to read the ROM image, or to replace an already existing ROM image (by the burn command). However, burning a new ROM image in case a previous image did not exist is not possible, nor is it possible to remove an existing ROM image</p>	Linux
mlxburn	Added the -fw_dir option which looks for a suitable firmware file in the given directory	Linux
	Support for generating a non-fail-safe image for ConnectX/ ConnectX-2, InfiniScale IV, and BridgeX devices	Linux
Debug tools	Updated the mlx2c utility	Linux
	Added the mget_temp utility which reads the temperature of the ConnectX/ ConnectX-2, InfiniScale IV, and BridgeX devices	Linux

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
2578580	Description: Fixed an issue that resulted in getting MVPD read errors from the mlxfwmanager during fast reboot.
	Keywords: mlxfwmanager, MVPD_READ4 failed, fast reboot
	Discovered in Version: 4.16.0
	Fixed in Release: 4.17.0
2628490	Description: Fixed inconsistent flashing of the firmware when using the IPMB service.
	Keywords: flint
	Discovered in Version: 4.16.0
	Fixed in Release: 4.17.0
2395589	<p>Description: Changed the flint "--activate" flag behavior to include a minimal delay of 1 second to avoid disconnections if the connected port is being activated.</p> <p>To use the "legacy" activation flow, use the "--activate_delay_sec 0" command.</p>

Internal Ref. No.	Issue
	Keywords: "--activate" flag, flint Discovered in Version: 4.16.0 Fixed in Release: 4.17.0
2494596	Description: Flint now supports the "--activate_delay_sec" flag which performs the activation on the newly burned firmware after the specified delay. Note: The burn flow will be locked after this command has been sent for a couple of minutes, until activation flow is done. Keywords: "--activate_delay_sec" flag, flint Discovered in Version: 4.16.0 Fixed in Release: 4.17.0
2443427	Description: Fixed an issue that resulted in "--json" flag not working with features that require a user confirmation. Note: Despite the fix, it is recommended to use the "--json" flag with the force flag set to yes. Keywords: mlxlink Discovered in Version: 4.16.0 Fixed in Release: 4.17.0
2071210	Description: mlxconfig query for the BOOT_INTERRUPT_DIS TLV shows a wrong value in the "current value" field. Keywords: mlxconfig Discovered in Version: 4.14.0-105 Fixed in Release: 4.17.0
2154936	Description: mst version returns an incorrect string: mst, MFT_VERSION_STR built on TOOLS_BUILD_TIME + Git SHA Hash: TOOLS_GIT_SHA Keywords: mst Discovered in Version: 4.14.2 Fixed in Release: 4.17.0
2224507	Description: mstflint is currently not in ConnectX-6 Lx adapter cards.

Internal Ref. No.	Issue
	Keywords: mstflint
	Discovered in Version: 4.15.0
	Fixed in Release: 4.17.0
2183083	Description: MFT tools do not support using combined short flags without a separation between them. For example: <ul style="list-style-type: none"> • Not recommended: -emc • Recommended: -e -m -c
	Keywords: Short flags
	Discovered in Version: 4.16.0
	Fixed in Release: 4.17.0
2391274	Description: mlxfwreset is not supported in SmartNIC devices.
	Keywords: mlxfwreset, SmartNIC devices.
	Discovered in Version: 4.16.3
	Fixed in Release: 4.17.0
2060223	Description: Performing a driver restart while burning the firmware results in firmware burning failure, and occasionally in device being inaccessible.
	Keywords: Firmware burning, driver restart
	Discovered in Version: 4.15.0
	Fixed in Release: 4.17.0
2200381	Description: CPLDUPDATE tool cannot work with both GPIO and firmware modes enabled on NVIDIA Spectrum-3 switches.
	Keywords: CPLDUPDATE
	Discovered in Version: 4.16.0
	Fixed in Release: 4.17.0
2439595	Description: Updated the following libraries versions: <ul style="list-style-type: none"> • OpenSSL to version 1.1.1i • Curl to version 7.75.0

Internal Ref. No.	Issue
	<ul style="list-style-type: none"> tcl to version 8.6.11 SQLite to version 3.33.0
	Keywords: Libraries
	Discovered in Version: 4.16.1
	Fixed in Release: 4.16.3
2400106	Description: Added support for signing kernel modules on Ubuntu/Debian
	Keywords: Secure boot
	Discovered in Version: 4.16.0
	Fixed in Release: 4.16.1
2297524	Description: Fixed an issue that caused lifecycle to be wrongly reported in ConnectX-6 adapter cards.
	Keywords: lifecycle, Connectx-6
	Discovered in Version: 4.14.0-105
	Fixed in Release: 4.16.0
2319179	Description: Fixed an issue that caused HMAC not to be written in livefish.
	Note: HMAC is now supported only from the Arm side and only if not in secure mode.
	Keywords: mlxlink
	Discovered in Version: 4.14.0-105
	Fixed in Release: 4.16.0
2084837	Description: Setting the speeds (50GbE and 100GbE) for the new devices (Connect-X 6 and above, Quantum switches and above) requires specifying the number of lanes for the speed: <code>mlxlink -d <dev> --speeds [50G_2X 50G_1X 100G_2X 100G_4X]</code> For PRBS mode, to work with PAM4 speeds, use the same speed naming for (50GbE, and 100GbE).
	Keywords: mlxlink
	Discovered in Version: 4.14.0-105
	Fixed in Release: 4.16.0
2125012	Description: In case a device enters the livefish mode and all the information on the flash including write-protected manufacturing information is lost, flint might not be able to recover the device.

Internal Ref. No.	Issue
	Keywords: flint Discovered in Version: 4.14.0-105 Fixed in Release: 4.16.0
2137820	Description: Running the flint query in parallel with the mst stop "-force" flag might cause the device to get into an undefined state and may require a power cycle to resolve the issue. Keywords: flint Discovered in Version: 4.14.2 Fixed in Release: 4.16.0
2151018	Description: Occasionally, when burning MFA2 using flint, it might get stuck if in the middle of the process mlxfwreset is executed. Keywords: MFA2, flint Discovered in Version: 4.15.0 Fixed in Release: 4.16.0
2234042	Description: Running CPLD on a remote device might take a very long time due to TCP transactions processing time. Keywords: cpldupdate Discovered in Version: 4.15.0 Fixed in Release: 4.16.0
2193807	Description: Cable firmware burning capability is not supported. Keywords: mlx cables Discovered in Version: 4.15.0 Fixed in Release: 4.16.0
2248709	Description: Burning tools cannot burn over mtusb interface, the tool will exit with the following error "mf object is NULL". Keywords: mstusb, burning tools Discovered in Version: 4.15.0 Fixed in Release: 4.16.0

Internal Ref. No.	Issue
2319984	Description: Fixed an issue that caused the margin scan to fail with the following message: Eye scan not completed.
	Keywords: mlxlink
	Discovered in Version: 4.15.0
	Fixed in Release: 4.16.0
1683637	Description: Fixed an issue that prevented mlxfwstress from turning ON stress types when two stress types conflicted with each other. In this case, using 'ALL' with 'on' operation resulted in mlxfwstress choosing to run only one of them.
	Keywords: mlxfwstress
	Discovered in Version: 4.15.0
	Fixed in Release: 4.16.0
2259628	Description: Wrong supported cable speed is displayed when using cable with P/N MCP2M00-A01A on a BlueField device.
	Keywords: BlueField, cables
	Discovered in Version: 4.15.0
	Fixed in Release: 4.16.0
2288076	Description: Fixed an issue that caused the device to be inaccessible for 3 minutes when applied bad tokens.
	Keywords: mlxconfig
	Discovered in Version: 4.15.1
	Fixed in Release: 4.16.0
2287949	Description: Hardware Security Module (HSM) capability is supported in secure firmware only and not in secure boot.
	Keywords: HSM, secure firmware, secure boot
	Discovered in Version: 4.15.0
	Fixed in Release: 4.16.0
2234042	Description: Fixed an issue that resulted in a long CPLD running time.
	Keywords: Installation

Internal Ref. No.	Issue
	Discovered in Version: 4.14.1
	Fixed in Release: 4.15.0
1885535	Description: Fixed deb installation in chroot environment.
	Keywords: Installation
	Discovered in Version: 4.14.1
	Fixed in Release: 4.15.0
2153427	Description: mlxlink cable commands do not work on FreeBSD 13.0-CURRENT OS.
	Keywords: mlxlink
	Discovered in Version: 4.14.2
	Fixed in Release: 4.15.0
2176654	Description: Fixed an issue that resulted in port split query failure on a switch with P/N MSN4600-CS2RO.
	Keywords: Port split
	Discovered in Version: 4.14.4
	Fixed in Release: 4.15.0
2123421	Description: Enabled the test mode while the port was disabled (unplugged cable).
	Keywords: mlxlink
	Discovered in Version: 4.14.0-105
	Fixed in Release: 4.14.2
2113431	Description: Fixed an issue that resulted in missing field (ob_leva) while setting the serdes_tx parameters.
	Keywords: mlxlink
	Discovered in Version: 4.14.0-105
	Fixed in Release: 4.14.1
1918749	Description: mlxlink tool displays a wrong speed when using ETH cables on ConnectX-6 adapter cards.
	Keywords: mlxlink
	Discovered in Version: 4.13.0
	Fixed in Release: 4.14.0-105
1895525	Description: If MFT is not installed via the standard installer, MLNX_WINMFT must be set manually to point to the MFT path.

Internal Ref. No.	Issue
	Keywords: MFT installation Discovered in Version: 4.13.0 Fixed in Release: 4.14.0-105
795705	Description: Fixed an issue that prevented mlxburn from reading the VPD on ppc64/ppc64le machines where the device shared the same B:D:F address with another PCI device on different PCI domains. Keywords: VPD, ppc64/ppc64le machines Discovered in Version: 4.4.0 Fixed in Release: 4.13.3
1608671/1523443	Description: mlxfwmanager "-download" command is currently not functional on PPC64/PPC64le and aarch64 platforms. Keywords: mlxfwmanager, PPC64/PPC64le/aarch64 Discovered in Version: 4.11.0 Fixed in Release: 4.13.0
1599465	Description: Fixed an issue that resulted in mlxfwreset failure when running it on an AMD processor. Keywords: mlxfwreset, AMD processor Discovered in Version: 4.11.0 Fixed in Release: 4.13.0
1655224	Description: Decreased mstflint query timeout from 80 seconds to 8 seconds. In case the tool does not get a response from the device after 8 seconds, the following error message is displayed: <i>"Cannot open Device: /dev/mst/mt4117_pciconf0. Resource unavailable".</i> Keywords: mstflint query Discovered in Version: 4.11.0 Fixed in Release: 4.12.0
1307423	Description: Execution of the mlxfwreset utility on a device with VFs configured may take longer than expected to be completed. Keywords: mlxfwreset Discovered in Version: 4.9.0 Fixed in Release: 4.11.0
1316844	Description: fwtrace is correctly not functional on PPC machines when the driver is loaded with firmware v 1x.22.1002. Keywords: fwtrace, PPC Discovered in Version: 4.9.0

Internal Ref. No.	Issue
	Fixed in Release: 4.11.0
1338958	Description: mlxfwreset is not supported in Socket Direct devices on Power platforms.
	Keywords: mlxfwreset, Power, Socket Direct
	Discovered in Version: 4.10.0
	Fixed in Release: 4.11.0
1406842	Description: MFT tools run slower on Bluefield devices. Firmware burning may take up to 20 minutes.
	Keywords: BlueField, firmware burn
	Discovered in Version: 4.10.0
	Fixed in Release: 4.11.0
1335391	Description: FW-reset for PPC Socket Direct is currently functional on Power9 InfiniBand setups only. The Minimum skiboot is: OPAL skiboot-v5.9-240-g081882690163
	Keywords: mlxfwreset
	Discovered in Version: 4.9.0
	Fixed in Release: 4.11.0
1356238	Description: Fixed an issue that caused the MFT tools' execution runtime in Windows OS to be longer than expected. e.g fastfwreset took up to 5 seconds to complete execution.
	Keywords: General, MFT tools
	Discovered in Version: 4.9.0
	Fixed in Release: 4.11.0
1056570/ 1058462	Description: Running the mlx cable tool in parallel on the same device (cable) may result in failure.
	Keywords: mlx cable
	Discovered in Version: 4.7.0
	Fixed in Release: 4.11.0
1321724	Description: Occasionally, when running mlxfwreset in Windows platforms, it may cause the device to malfunction during the reset process.

Internal Ref. No.	Issue
	Keywords: mlxfwreset
	Discovered in Version: 4.9.0
	Fixed in Release: 4.10.0
1315138	Description: Fixed an issue that caused mlxburn to fail generating a binary image in Windows.
	Keywords: Image generation
	Discovered in Version: 4.9.0
	Fixed in Release: 4.10.0
1350622	Description: Fixed an issue that prevented the driver from starting on a PPC platforms when used the mlxfwreset tool.
	Keywords: mlxfwreset
	Discovered in Version: 4.9.0
	Fixed in Release: 4.10.0
1213983	Description: Connect-IB function per port (FPP_EB) is not exposed at mlxconfig.
	Keywords: mlxfwreset, Connect-IB
	Discovered in Version: 4.7.0
	Fixed in Release: 4.9.0
540511	Description: If an unexpected shutdown occurs after running the firmware update package (UPMF) in Windows, 'mst status' may not show any devices when the machine comes up.
	Keywords: mst
	Discovered in Version: 4.0.0
	Fixed in Release: 4.8.0
554872	Description: FreeBSD PCI access API is currently not supported.
	Keywords: mlxburn
	Discovered in Release: 4.0.0
	Fixed in Release: 4.8.0

Internal Ref. No.	Issue
1064918/ 1069102	Description: mlxfwreset does not load the firmware properly on a Socket-Direct card.
	Keywords: mlxfwreset
	Discovered in Release: 4.7.0
	Fixed in Release: 4.8.0
1041544/ 1041545	Description: When the port is set with NO FEC, the Raw Errors Counters will always show 0.
	Keywords: mlxlink, Raw Errors Counters
	Discovered in Release: 4.7.0
	Fixed in Release: 4.8.0
1097425	Description: mlxfwmanager does not handle Socket Direct adapters correctly.
	Keywords: mlxfwmanager
	Discovered in Release: 4.7.0
	Fixed in Release: 4.8.0
676539	Description: mlxuptime and mget_temp are not working against INBAND ConnectX-4/ ConnectX-4 Lx devices.
	Keywords: mlxuptime
	Discovered in Release: 4.4.0
	Fixed in Release: 4.7.0
955525	Description: Image generation fails when generating a ConnectX-5 image on FreeBSD12-CURRENT.
	Keywords: ConnectX-5, image generation, FreeBSD12-CURRENT
	Discovered in Release: 4.6.0
	Fixed in Release: 4.7.0
907531	Description: mlxfwreset is not functional on MultiHost and Socket Direct NICs.
	Keywords: mlxfwreset
	Discovered in Release: 4.6.0

Internal Ref. No.	Issue
	Fixed in Release: 4.7.0
969322/ 969566	Description: mlxfwreset may fail to reset the device on Ubuntu PPC64LE systems when multiple kernels are installed.
	Keywords: kernel module, mlxfwreset, Ubuntu PPC64LE
	Discovered in Release: 4.6.0
	Fixed in Release: 4.7.0
759915/ 778296/ 854084/ 795109/ 759015	Description: Segmentation fault may occur in fwtrace on RedHat 6.5 and 6.7 systems.
	Keywords: fwtrace
	Discovered in Release: 4.4.0
	Fixed in Release: 4.7.0
795226/ 795657/ 862607	Description: Occasionally, MFT tools (driver mode) do not function after running mlxfwreset in PowerPC machines.
	Keywords: mlxfwreset
	Discovered in Release: 4.4.0
	Fixed in Release: 4.6.0
795028/ 795705	Description: mlxburn fails to read VPD on machines where the device shares the same B:D:F address with another PCI device on different PCI domains
	Keywords: mlxburn
	Discovered in Release: 4.4.0
	Fixed in Release: 4.6.0
385113	Description: Reading the VPD using the “-vpd_rw” flag or programing the VPD may take up to 5 mins.
	Keywords: mlxburn
	Discovered in Release: 3.7.0
	Fixed in Release: 4.6.0
795756/ 795916	Description: mlxfwreset disables and enables all Mellanox devices' Network Interfaces when resetting the firmware on a device that at least one of its network interfaces is up.

Internal Ref. No.	Issue
	Keywords: mlxfwreset
	Discovered in Release: 4.4.0
	Fixed in Release: 4.5.0
795479/ 795521	Description: Running mlxfwreset against OEM devices may enter the device to a undefined state.
	Keywords: mlxfwreset
	Discovered in Release: 4.4.0
	Fixed in Release: 4.5.0
697509	Description: PPTT and PPRT registers are not supported in switches.
	Keywords: mlxreg
	Discovered in Release: 4.3.0
	Fixed in Release: 4.5.0
757651/ 778451	Description: Fixed an issue causing the mlx cables tool to show wrong temperature value when querying the cable several times in loop.
	Keywords: mlx cables
	Discovered in Release: 4.4.0
	Fixed in Release: 4.5.0

Notice

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

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

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

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

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

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

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

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

Trademarks

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

Copyright

© 2022 NVIDIA Corporation & affiliates. All Rights Reserved.

