Mellanox Firmware Tools (MFT) User Manual

Rev 3.2
Software ver. 4.9.0

NOTE:
THIS HARDWARE, SOFTWARE OR TEST SUITE PRODUCT ("PRODUCT(S)") AND ITS RELATED DOCUMENTATION ARE PROVIDED BY MELLANOX TECHNOLOGIES "AS-IS" WITH ALL FAULTS OF ANY KIND AND SOLELY FOR THE PURPOSE OF AIDING THE CUSTOMER IN TESTING APPLICATIONS THAT USE THE PRODUCTS IN DESIGNATED SOLUTIONS. THE CUSTOMER'S MANUFACTURING TEST ENVIRONMENT HAS NOT MET THE STANDARDS SET BY MELLANOX TECHNOLOGIES TO FULLY QUALIFY THE PRODUCT(S) AND/OR THE SYSTEM USING IT. THEREFORE, MELLANOX TECHNOLOGIES CANNOT AND DOES NOT GUARANTEE OR WARRANT THAT THE PRODUCTS WILL OPERATE WITH THE HIGHEST QUALITY. ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT ARE DISCLAIMED. IN NO EVENT SHALL MELLANOX BE LIABLE TO CUSTOMER OR ANY THIRD PARTIES FOR ANY DIRECT, INDIRECT, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES OF ANY KIND (INCLUDING, BUT NOT LIMITED TO, PAYMENT FOR PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY FROM THE USE OF THE PRODUCT(S) AND RELATED DOCUMENTATION EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Mellanox Technologies
350 Oakmead Parkway Suite 100
Sunnyvale, CA 94085
U.S.A.
www.mellanox.com
Tel: (408) 970-3400
Fax: (408) 970-3403

Table of Content

List of Tables

List of Figures

Document Revision History

Table 1: Revision History Table (Sheet 1 of 12)

Revision	Date	Description

3.2	March 1, 2018	• Added the following sections:
		  • Section 2.4.9.2, “For 5th Generation Devices”, on page 93
		• Updated the following sections:
		  • Section 2.3.3.1, “Querying the Device Configuration”, on page 48
		  • Section 2.3.13, “mlxconfig create_conf Command”, on page 57
		  • Section 2.4.18.1, “Signing Binary Image Files”, on page 100
		  • Section 3.6.1, “mget_temp Usage”, on page 126
		  • Section 3.15.1, “mlxlink Usage”, on page 141
		  • Section 7, “Supported Configurations and their Parameters”, on page 58: Added the following parameters:
		SW_RECOVERY_ON_ERRORS
		RESET_WITH_HOST_ON_ERRORS
		IBM_TUNNELED_ATOMIC_EN
		EXP_ROM_PXE_ENABLE
		EXP_ROM_UEFI_x86_ENABLE
		EXP_ROM_UEFI_ARM_ENABLE
		HOST_CHAINING_MODE
		HOST_CHAINING_DESCRIPTORS
		HOST_CHAINING_TOTAL_BUFFER_SIZE
		  • Section 2.4.17.1, “Using Secure Host”, on page 98

3.1	October 30, 2017	• Added the following sections:
		  • Section 2.4.18.3, “Setting a "Forbidden Versions" Section in a Binary Image File”, on page 101
		  • Appendix E: “Generating Firmware Secure and NV LifeCycle Configurations Files,” on page 183
		  • E.1 “Create Forbidden Versions Binary File,” on page 183
		  • E.2 “Create Tokens for Secure Firmware and NV LifeCycle,” on page 183

Table 1: Revision History Table (Sheet 2 of 12)

Revision	Date	Description

3.1	October 30, 2017	• Updated the following sections:
		  • Section 2.1.1.1.1, "mst Commands and Switches Description - Linux", on page 26
		  • Section 2.2, "mlxfwmanager - Firmware Update and Query Tool", on page 34
		  • Section 2.2.6.2, "UPMF Generation Example", on page 44
		  • Section 2.3.13, "mlxconfig create_conf Command", on page 57
		  • Section 2.3.15, "Supported Configurations and their Parameters", on page 58: added Global Conf parameter
		  • Section 2.6.4, "mlxfwreset for Multi-Host NICs", on page 110: added Socket Direct
		  • Section 2.8.2, "mlx_fpga Synopsis", on page 114
		  • Section 2.8.3.1, "Adding FPGA mst Device Interface", on page 114
		  • Section 3.8, "mlxdump Utility", on page 128
		  • Section 3.8.1, "mlxdump Usage", on page 128
		  • Section 5.4, "Firmware Burning Related Issues", on page 158
		  • Section 3.15, "mlxlink Utility", on page 141: Added "`--show_ber_monitor`" flag
		• Updated Table 7, "Supported Configurations and their Parameters," on page 58
		  • Added the following Features/Parameters and their configuration:
		  • BOOT_LACP_DIS
		  • IP_OVER_VXLAN_EN
		  • IP_OVER_VXLAN_PORT
		  • HCA Conf
		  • ROM UEFI Conf
		  • Removed DCR_CNACK_BUFFER_SIZE

Table 1: Revision History Table (Sheet 3 of 12)

Revision	Date	Description
3.0	June 29, 2017	• Added the following sections: • Section 2.3.10, "Generating an XML Template for the Configurations", on page 55 • Section 2.3.11, "mlxconfig xml2raw Command", on page 56 • Section 2.3.12, "mlxconfig xml2bin Command", on page 56 • Section 2.3.13, "mlxconfig create_conf Command", on page 57 • Section 2.3.14, "mlxconfig apply Command", on page 57 • Section 2.4.18, "Secure Firmware Update", on page 100 • Section 2.4.18.1, "Signing Binary Image Files", on page 100 • Section 2.4.18.2, "Setting a "Public Keys" Section in a Binary Image File", on page 101 • Section 2.4.18.4, "Secure Firmware Implications on Burning Tools", on page 101 • Section 2.6.4, "mlxfwreset for Multi-Host NICs", on page 110 • Section 4.3.2, "MTUSB Cable Board", on page 152 • Section 4.3.3, "Cable Firmware Upgrade with In Service Firmware Update (ISFU)", on page 154 • Section , "5th Generation Devices Examples:", on page 135: added CMAS examples • Section 3.15, "mlxlink Utility", on page 141

Table 1: Revision History Table (Sheet 4 of 12)

Revision	Date	Description
3.0	June 29, 2017	• Updated the following sections: • Section 2.2.1, "mlxfwmanager Synopsis", on page 34: added [--no_fw_ctrl] • Section 2.3.2, "mlxconfig Synopsis", on page 47: added [-p|--private_key], [-u|--key_uuid], , [xml2bin], [create_conf] and [apply] • Section 2.4.1.1, "Switches Options", on page 77: added [--no_fw_ctrl] • Section 2.2.4.1, "Downloading Firmware Images and Firmware Update Packages", on page 40: updated the examples output • Section 2.7.2, "mlxphyburn Synopsis", on page 113: updated the synopsis format • Section 2.8.2, "mlx_fpga Synopsis", on page 114 • Section 3.1.1, "fwtrace Usage", on page 119: added the "-n|--snapshot" flag and added ConnectX-5 iRISC names • Section 3.12.1, "wqdump Usage", on page 133: added ConnectX-5 sources to flag "--source ContextType" • Section 4.3, "mlxcables - Mellanox Cables Tool", on page 146 • Section 4.3.1, "mlxcables Synopsis", on page 146: updated the synopsis output • Section 5.4, "Firmware Burning Related Issues", on page 158: added Secure Firmware related issues • Section 2.3.15, "Supported Configurations and their Parameters", on page 58 • Section 2.4.3, "Querying the Firmware Image", on page 82 • Section 3.3, "mstdump Utility", on page 124 • Section 3.11, "mlxuptime Utility", on page 132

Table 1: Revision History Table (Sheet 5 of 12)

Revision	Date	Description

2.90	January 31, 2017	• Added the following sections:
		  • Section 2.5.4.2, "ConnectX-5® Examples", on page 107
		  • Section , "5th Generation Devices Examples:", on page 135
		• Updated the following sections:
		  • Section 1.2, "MFT Access to Hardware Devices", on page 22: added ConnectX-5
		  • Section 1.3, "MFT Installation", on page 24: updated the MFT Installation table
		  • Section 2.2, "mlxfwmanager - Firmware Update and Query Tool", on page 34: added a note related to "mlxfw-manager_pci" tool being deprecated.
		  • Section 2.3.2, "mlxconfig Synopsis", on page 47: replaced the "show_default" flag with "enable_verbosity" and updated the flags description
		  • Section 2.3.15, "Supported Configurations and their Parameters", on page 58: added new parameter NON_PREFETCHABLE_PF_BAR
		  • Section 2.3.9, "mlxconfig Backup Command", on page 54: updated the command's output
		  • Section 2.4.1.1, "Switches Options", on page 77: added the "-use_dev_rom" parameter
		  • Section 2.8.2, "mlx_fpga Synopsis", on page 114: updated the synopsis
		  • Section 2.8.3.3, "Loading the Tool", on page 116: added -the "-user" parameter

Table 1: Revision History Table (Sheet 6 of 12)

Revision	Date	Description
2.8	September 30, 2016	Added the following sections: • Section 2.4.14, "Querying Flash Parameters", on page 95 • Section 2.8.3.3, "Loading the Tool", on page 116 • Section 2.8.3.5, "Updating the FPGA Image", on page 116 Updated the following sections: • Section 2.3.2, "mlxconfig Synopsis", on page 47 • Section 2.3.15, "Supported Configurations and their Parameters", on page 58 • Section 2.5.3.1, "Mellanox Connect-IB®, Switch-IB™, Switch-IB™ 2, Spectrum, ConnectX®-4 and ConnectX®-4 Lx Initial Burning Options", on page 106 • Section 2.8, "mlx_fpga - Burning and Debugging Tool for Mellanox Devices with FPGA", on page 113 • Section 2.8.1, "Tool Requirements", on page 113 • Section 2.8.2, "mlx_fpga Synopsis", on page 114 • Section 2.8.3, "Examples of mlx_fpga Usage", on page 114 • Section 2.8.3.1, "Adding FPGA mst Device Interface", on page 114 • Section 2.8.3.2, "Burning the FPGA's Flash Device using the mlx_fpga Burning Tool", on page 115 • Section 2.8.3.4, "Debugging the Tool", on page 116 • Section 3.3.1, "mstdump Usage", on page 124 • Section 3.6.1, "mget_temp Usage", on page 126 • Section 3.10.1, "pckt_drop Usage", on page 131 • Section 4.1.2.1, "Local PCI Cables", on page 144 • Section 4.3.1, "mlxcables Synopsis", on page 146

Table 1: Revision History Table (Sheet 7 of 12)

Revision	Date	Description
2.7	May 2016	Added the following sections: • Section 2.1.2, "Running mst in an Environment without a kernel", on page 29 • Section 2.3.9, "mlxconfig Backup Command", on page 54 • Section 4, "Cables Utilities", on page 144 • Section 4.1, "Cables Discovery", on page 144 • Section 4.2, "Working with Cables", on page 145 • Section 4.3, "mlxcables - Mellanox Cables Tool", on page 146 Updated the following sections: • Section 2.1.3.1.1, "mst Commands and Switches Description - Windows", on page 30 • Section 2.1.4.1.1, "Commands and Switches Description - FreeBSD", on page 32 • Section 2.1.5.1.1, "Commands and Switches Description - VMware", on page 33 • Section 2.2.1, "mlxfwmanager Synopsis", on page 34 • Section 2.2.4.1, "Downloading Firmware Images and Firmware Update Packages", on page 40 • Section 2.3.2, "mlxconfig Synopsis", on page 47 • Section 2.3.15, "Supported Configurations and their Parameters", on page 58 • Section 2.8.2, "mlx_fpga Synopsis", on page 114 • Section 5.4, "Firmware Burning Related Issues", on page 158
2.6	January 2016	Added the following sections: • Section 2.4.15, "Firmware Time-stamping for Multi-host Environment", on page 95 • Section 2.4.5, "Performing Checksum Calculation on Image/Device", on page 83 • Section 3.14, "mlxreg Utility", on page 138 • D.4 "Spectrum™ or Switch-IB™ 2 Switch Systems," on page 180 Updated the following sections: • Section 1.2, "MFT Access to Hardware Devices", on page 22 • Section 2.3, "mlxconfig - Changing Device Configuration Tool", on page 47 • Section 2.5, "mlxburn - Firmware Image Generator and Burner", on page 102

Table 1: Revision History Table (Sheet 8 of 12)

Revision	Date	Description
2.5	September 2015	Added the following sections: • Section 2.3.6, "Using mlxconfig to Set SR-IOV Parameters", on page 52 • Section 2.3.7, "Using mlxconfig to Set preboot Settings", on page 52 • Section 2.3.8, "mlxconfig Raw Configuration Files", on page 53 • Section 2.9, "cpldupdate - Tool for Programming On-Board CPLDs on Mellanox Devices", on page 117 • B.3.6 "Switch Reprogramming through I2C Port," on page 169 Updated the following sections: • Section 2.3, "mlxconfig - Changing Device Configuration Tool", on page 47 • Section 3.1.1, "fwtrace Usage", on page 119 • Section 3.2.1, "itrace Usage", on page 122 • Section 3.7.1, "mlxtrace Usage", on page 127 • Section 2.5.2, "Customizing Firmware", on page 103 • Section B, "Remote Access to Mellanox Devices", on page 164

Table 1: Revision History Table (Sheet 9 of 12)

Revision	Date	Description
2.4	June 2015	Added the following sections: • Section 2.8, "mlx_fpga - Burning and Debugging Tool for Mellanox Devices with FPGA", on page 113 • Section D.4, "Spectrum™ or Switch-IB™ 2 Switch Systems", on page 180 • Section 2.3.6, "Using mlxconfig to Set SR-IOV Parameters", on page 52 • Section 2.3.8, "mlxconfig Raw Configuration Files", on page 53 • Section 2.9, "cpldupdate - Tool for Programming On-Board CPLDs on Mellanox Devices", on page 117 Updated the following sections: • Section 1.3, "MFT Installation", on page 24 • Section 2.1, "Mellanox Software Tools (mst) Service", on page 26 • Section 2.5, "mlxburn - Firmware Image Generator and Burner", on page 102 • Section 2.6, "mlxfwreset - Loading Firmware on 5th Generation Devices Tool", on page 109 • Section 2.6, "mlxfwreset - Loading Firmware on 5th Generation Devices Tool", on page 109 • Section 3.2, "itrace Utility", on page 121 • Section 3.7, "mlxtrace Utility", on page 126 • Section 3.12, "wqdump Utility", on page 132 • Section 5, "Troubleshooting", on page 156 • Section D.1.2, "PCI Vital Product Data (VPD)", on page 175 • 2.3 "mlxconfig - Changing Device Configuration Tool," on page 47
2.3	January 2015	Added the following sections: • Section 2.7, "mlxphyburn – Burning Tool for Externally Managed PHY", on page 112 • Section D.3, "Switch-IB™ Switch System", on page 178 Updated the following sections: • 2.3.15 "Supported Configurations and their Parameters," on page 58 • 2.3.3 "Examples of mlxconfig Usage," on page 48 • Section 3.1, "fwtrace Utility", on page 119 • Section 3.2.1, "itrace Usage", on page 122 • Section 2.2.6.2, "UPMF Generation Example", on page 44

Table 1: Revision History Table (Sheet 10 of 12)

Revision	Date	Description
2.2	August 2014	Added the following sections: • Section 2.6, “mlxfwreset - Loading Firmware on 5th Generation Devices Tool”, on page 109 • Section 2.1, “Mellanox Software Tools (mst) Service”, on page 26 • Section C, “Booting HCA Device in Livefish Mode”, on page 173 Updated the following sections: • Section 2.6, “mlxfwreset - Loading Firmware on 5th Generation Devices Tool”, on page 109 • Section C, “Booting HCA Device in Livefish Mode”, on page 173 • Section 2.4.1, “flint Synopsis”, on page 76
2.1	May 2014	Added the following sections: • Section 2.2, “mlxfwmanager - Firmware Update and Query Tool”, on page 34 • Section 2.4.17, “Secure Host”, on page 97 Updated the following sections: • Section 2.6, “mlxfwreset - Loading Firmware on 5th Generation Devices Tool”, on page 109 • Section 2.3.15, “Supported Configurations and their Parameters”, on page 58 • Section 2.7.3.1, “Burn Example”, on page 113 • Section , “Example (VIBs installation):”, on page 25
2.0	February 2014	Removed the sub-section “How to Run flint in VMware” Removed the sub-section “How to run mstdump in VMware” Updated the following sections: • Section , “Example (VIBs installation):”, on page 25 • Section , “”, on page 25 • Section 3.12, “wqdump Utility”, on page 132

Table 1: Revision History Table (Sheet 11 of 12)

Revision	Date	Description
1.90	December 2013	Removed the -qq flag from the document Removed sub-section “On Pre-ConnectX Devices” Added the following sections: • Section 3.12, “wqdump Utility”, on page 132 Updated the following sections: • Section 1.3, “MFT Installation”, on page 24 • Section 2.1, “Mellanox Software Tools (mst) Service”, on page 26 • Section , “Example (VIBs installation):”, on page 25 • Section 2.5.3, “mlxburn Synopsis”, on page 104 • Section 2.5.4.5, “Connect-IB® Examples”, on page 107 • Section , “InfiniScale IV Switch Examples”, on page 117 • Section 2.4.1.2, “Command Parameters”, on page 79 • Section 2.4.9, “Disabling/Enabling Access to the Hardware”, on page 92 • Section 2.4.16, “Flint/mlxburn Limitations”, on page 97 • Section 2.6, “mlxfwreset - Loading Firmware on 5th Generation Devices Tool”, on page 109 • Section 2.2.2, “Querying the Device”, on page 35 • Section 2.2.3, “Updating the Device”, on page 37 • Section 3.4, “mlxi2c Utility”, on page 124 • Section 3.5.1, “i2c Usage (Advance Users)”, on page 125 • Section 3.7, “mlxtrace Utility”, on page 126 • Section 3.12, “wqdump Utility”, on page 132 • Section D.1.3, “Burning a New Connect-IB® Device”, on page 175
1.80	October 2013	Updated section Section 2.4.1.2, “Command Parameters”, on page 79 added a Connect-IB™ Expansion ROM command limitation note.
1.80	July 2013	Reorganized the Firmware Tools and Utilities section. Added the following sections: • Section 2.6, “mlxfwreset - Loading Firmware on 5th Generation Devices Tool”, on page 109 • Section 3.1, “fwtrace Utility”, on page 119 • Section 3.11, “mlxuptime Utility”, on page 132 • B.5 “Accessing Remote InfiniBand Device by Direct Route MADs,” on page 171 • B.5 “Accessing Remote InfiniBand Device by Direct Route MADs,” on page 171 • Appendix C: “Booting HCA Device in Livefish Mode,” on page 173

Table 1: Revision History Table (Sheet 12 of 12)

Revision	Date	Description
1.70	April 2013	Added the following sections: • Section 2.5.3.1, "Mellanox Connect-IB®, Switch-IB™, Switch-IB™ 2, Spectrum, ConnectX®-4 and ConnectX®-4 Lx Initial Burning Options", on page 106 • Section 2.5.4.5, "Connect-IB® Examples", on page 107 • Section 3.2, "itrace Utility", on page 121 • Section 3.8, "mlxdump Utility", on page 128 • Section 3.7, "mlxtrace Utility", on page 126 • Section 3.9, "mlxmcg Utility", on page 129 • Section 3.10, "pckt_drop Utility", on page 131 • Appendix B.4, "Remote Access to Device by Sockets" Removed the following sections: • Section 1.2, "Software Prerequisites" • Section 2.1.4, "Exit Return Values" Updated the following sections: • Section 2.5.3, "mlxburn Synopsis", on page 104 • Section 2.5.4, "Examples of mlxburn Usage", on page 106 • Section 2.4.1, "flint Synopsis", on page 76

About this Manual

The document describes MFT v4.9.0 features, tools content and configuration.

Intended Audience

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

Common Abbreviations and Acronyms

Table 2: Common Abbreviations and Acronyms

Term	Description
MFT	Mellanox Firmware tools
MST	Mellanox Software tools and it's 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
ISFU	In Service Firmware Update
4th Generation ICs/Group I of ICs	Contains the following devices: • ConnectX®-3 • ConnectX®-3 Pro • SwitchX® • SwitchX®-2
5th Generation ICs/Group II of ICs	Contains the following devices: • Connect-IB® • Switch-IB™ • Switch-IB™ 2 • Spectrum™ • ConnectX®-4 • ConnectX®-4 Lx • ConnectX®-5 • ConnectX®-5 Ex

Reference Documents and Downloads

Table 3: 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
Mellanox OFED driver (for Linux)	http://www.mellanox.com > Products > Software > InfiniBand/VPI Drivers > Linux Driver.
Mellanox WinOF/WinOF-2 driver (for Windows)	http://www.mellanox.com > Products > Software > InfiniBand/VPI Drivers > Windows Driver.
Mellanox OFED (for ESXi)	http://www.mellanox.com > Products > Software > InfiniBand/VPI Drivers > VMware Driver.
Mellanox OFED (for FreeBSD)	http://www.mellanox.com > Products > Software > InfiniBand/VPI Drivers > FreeBSD Driver.

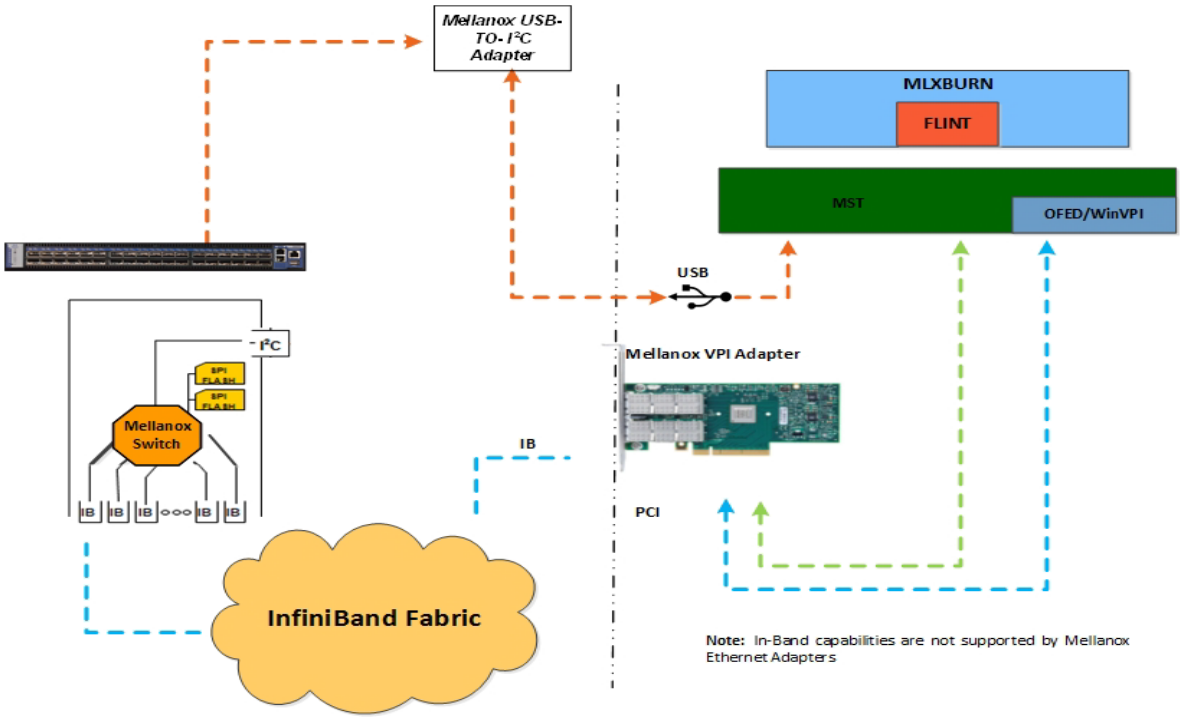
1 Introduction

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

- Generating a standard or customized Mellanox 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.

Figure 1: Mellanox Firmware Tools – A Scheme of Operation



1.1 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.

1.2 MFT Access to Hardware Devices

Table 4 lists the Mellanox devices supported by MFT, the supporting tools, and the access methods to these devices.

Table 4: Supported Mellanox Devices

Device Type	Product Name	HW Access Method		
		PCI	I2C	In-Band
HCA (InfiniBand)	Connect-IB®	V	V	V
VPI Network Adapter	ConnectX®-3	V	V	V
	ConnectX®-3 Pro	V	V	V
	ConnectX®-4	V	V	V
	ConnectX®-5	V	V	V
	ConnectX®-5 Ex	V	V	V
Ethernet Adapter (NIC)	ConnectX®-3 EN	V	V	
	ConnectX®-4 Lx	V	V	
Switch	SwitchX®-2	V[1]	V	V
	SwitchX®	V[1]	V	V
	Switch-IB™	V[1]	V	V
	Switch-IB™ 2	V[1]	V	V
	Spectrum™	V	V	

1. For managed switch products only.

MFT tools access Mellanox devices via the PCI Express interface, via a USB to I2C adapter (Mellanox 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.

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 Appendix B,“Remote Access to Mellanox Devices,” on page 164 for instructions on how to obtain the device LID

- **Via PCI user level:** <bus:dev.fn>

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

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

1.3 MFT Installation

Download the relevant MFT package for your OS from the Mellanox Management Tools webpage: http://www.mellanox.com/products/management_tools.php and continue as described in table bellow according your OS.

Table 5: MFT Installation

OS	Install	Uninstall
Linux	1. Untar the downloaded package 2. Allow packaging of bins to self executing file. 3. Run 'install.sh' **For OEM only:** 'install.sh --oem' 4. Start the mst driver by running: mst start **NOTE:** It is possible to customize some installation parameters (such as the target installation path). Run '`install.sh --help`' for details.	Uninstall MFT on Linux by running the following command: `mft_uninstall.sh`
Windows	The installation is EXE based: 1. Double click the EXE file and follow the instructions presented by the installation wizard.	1. Go to Add or remove programs. 2. Remove WinMFT64 depending on the platform type.
FreeBSD	1. Untar the downloaded package. 2. Run "`install.sh`"	Uninstall MFT on FreeBSD, run the following command: `mft_uninstall.sh`

VMware	1. Install the package. Run:
	`# esxcli software vib install -v <MST Vib>`
	`# esxcli software vib install -v <MFT Vib>`
	
	NOTE: For VIBs installation examples, please see below.
	
	2. Reboot system.
	3. Start the mst driver. Run:
	`# /opt/mellanox/bin/mst start`
		1. Uninstall the package. Run:
		`# esxcli software vib remove -n mft`
		2. Uninstall the mst:
		• VMKlinux:
		`# esxcli software vib remove -n net-mft`
		• Native:
		`# esxcli software vib remove -n nmst`
		3. Reboot system.

Example (VIBs installation):

- VMK:

```
esxcli software vib install -v /tmp/net-mst_4.6.0.22-1OEM.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-4.6.0.22-1OEM.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
```

The MFT tools are not located in the default path. In order to run any MFT tool either:

- Enter the full path. For example: `/opt/mellanox/bin/flint`

OR

- Add MFT path to the default system path by running: `export PATH=$PATH:/opt/mellanox/bin`. Please note, the path is temporary and will hold only until reboot

2 Firmware Generation, Configuration, and Update Tools

2.1 Mellanox Software Tools (mst) Service

2.1.1 Linux

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

2.1.1.1 mst Synopsis - Linux

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

2.1.1.1.1 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 Mellanox 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 and you can invoke other Mellanox tools like Infiniburn or tdevmon. You can configure the start command by edit the configuration file: /etc/mft/mst.conf, for example you can rename you devices. Options: • `--with_msix`: Create the msix device. • `--with_unknown`: Do not check if the device ID is supported. • `--with_i2cdev`: Create Embedded I2C master • `--with_fpga`: Create MST device for the attached FPGA card (Access via Driver) • `--with_fpga_fw_access`: Create an extended MST device for the attached FPGA (Access via Firmware)
`mst stop [--force]`	Stop Mellanox MST driver service, remove all special files/directories and unload kernel modules. Options: • `--force`: Force try to stop mst driver even if it's in use.
`mst restart [--with_msix] [--with_unknown] [--with_i2cdev] [--with_lpcdev] [--with_fpga] [--with_fpga_fw_access]`	Just like "`mst stop`" followed by "`mst start [--with_msix] [--with_unknown] [--with_i2cdev] [--with_lpcdev] [--with_fpga] [--with_fpga_fw_access]`"
`mst server start [port]`	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 add [OPTIONS] [local_hca_id] [local_hca_port]`	Add devices found in the IB fabric for inband access. Requires OFED installation and an active IB link. If local_hca_id and local_hca_port are given, the IB subnet connected to the given port is scanned. Otherwise, the default subnet is scanned. **Options:** • `--discover-tool <discover-tool>`: The tool that is used to discover the fabric. **Supported tools:** ibnetdiscover, ibdiagnet. default: ibdiagnet • `--add-non-mlnx`: Add non Mellanox nodes. • `--topo-file <topology-file>`: A prepared topology file which describes the fabric. For ibnetdiscover: provide an output of the tool. For ibdiagnet: provide LST file that ibdiagnet generates. • `--use-ibdr`: Access by direct route MADs. Available only when using ibnetdiscover tool, for SwitchX and ConnectIB devices. **NOTE**: if a topology file is specified, device are taken from it. Otherwise, a discover tool is run to discover the fabric.
`mst ib del`	Remove all inband devices.
`mst cable add [OPTIONS] [params]`	Add the cable that are connected to the devices, which support the MCIA access register. 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 Mellanox devices Options: • -v run with high verbosity level (print more info on each device)
`mst save`	Save PCI configuration headers in directory /var/mst_pci.
`mst load`	Load PCI configuration headers from directory /var/mst_pci.
`mst version`	Print the version info

2.1.1.1.2 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
```

Table 6: mst start Supported OPCODES

OPCODES	Definition	Description

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

2.1.1.1.3 Examples of mst Usage - Linux

➢ ***To start Mellanox 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:***

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

➢ ***To print the current status of Mellanox devices:***

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

MST devices:
------------
/dev/mst/mt4099_pciconf0         - PCI configuration cycles access.
                                   domain:bus:dev.fn=0000:0b:00.0 addr.reg=88 data.reg=92
                                   Chip revision is: 01
/dev/mst/mt4099_pci_cr0          - PCI direct access.
                                   domain:bus:dev.fn=0000:0b:00.0 bar=0xd2600000
                                   size=0x100000
                                   Chip revision is: 01
/dev/mst/mtusb-1                 - USB to I2C adapter as I2C master
                                   iSerial = 0x1683
```

➢ ***To show the devices status with detailed information:***

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

ConnectIB(rev:0)        /dev/mst/mt4113_pciconf0      0b:00.0   mlx5_2    net-ib4,           -1
                                                                          net-ib5
ConnectX3(rev:1)        /dev/mst/mt4099_pciconf0
ConnectX3(rev:1)        /dev/mst/mt4099_pci_cr0       0e:00.0   mlx4_0    net-ib0,           -1
                                                                          net-ib1
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
```

2.1.2 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                GuidsNumber
Base GUID:           7cfe90030029205e        4
Base MAC:            00007cfe9029205e        4
Image VSD:
Device VSD:
PSID:                MT_2190110032
```

2.1.3 Windows

2.1.3.1 mst Synopsis - Windows

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

2.1.3.1.1 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 <host-name>[: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 <host-name>[: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 cable that are connected to the devices, which support the MCIA access register. 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 Mellanox devices
`mst version`	Print the version info
`mst ib add [OPTIONS] [local_hca_id] [local_hca_port] [lst-file]`	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, ibnet-discover tool is run to discover the fabric. **Options:** • `--discover-tool <discover-tool>`: The tool used to discover the fabric. **Supported tools:** ibnetdiscover, ibdiagnet. default: ibnetdiscover **NOTE**: The discover tool argument is intended only for parsing purposes, thus you need to specify an lst-file with it. • `--add-non-mlnx`: Add non Mellanox nodes. • `--use-ibdr`: Access by direct route MADs. Available only when using ibnetdiscover tool, for SwitchX and ConnectIB devices. • --no-format-check: Do not check the format of the given local_hca_id. The expected format of the local_hca_id is: ibv_device[0-9]+. • `--topo-file <topology-file>`: A prepared topology file which describes the fabric. For ibnetdiscover: provide an output of the tool. For ibdiagnet: provide and lst-file that ibdiagnet generates. **NOTE**: If a topology file is specified, the devices are taken from it. Otherwise, a discover tool is run to discover the fabric.
`mst help`	Print this help information.

2.1.3.1.2 Examples of mst Usage - Windows

➢ *To print the current status of Mellanox devices:*

```
# mst status
MST devices:
------------
mt4115_pciconf0
mtusb-1
mtusb-2
```

➢ *To show the devices status with detailed information:*

```
# mst status -v

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

2.1.4 FreeBSD

2.1.4.1 mst Synopsis - FreeBSD

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

2.1.4.1.1 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

2.1.4.1.2 Examples of mst Usage - FreeBSD

➢ *To print the current status of Mellanox devices:*

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

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

2.1.5 VMware ESXi

2.1.5.1 mst Synopsis - VMware

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

2.1.5.1.1 Commands and Switches Description - VMware

mst start	Create special files that represent Mellanox devices in directory/dev. Load appropriate modules. After successfully completing this command, the mst driver will be ready to work.
mst stop	Stop Mellanox 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 Mellanox devices Options: -v run with a high verbosity level (print more info on each device)
mst version	Print the version info

2.1.5.1.2 Examples of mst Usage - VMware

- ***To print the current status of Mellanox 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       RD    NET                   NU
                                                     MA                          MA
```

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 B.2 “In-Band Access to Multiple IB Subnets,” on page 165, B.5 “Accessing Remote InfiniBand Device by Direct Route MADs,” on page 171 and Appendix B.4,“Remote Access to Device by Sockets,” on page 169

2.2 mlxfwmanager - Firmware Update and Query Tool

The mlxfwmanager is a Mellanox firmware update and query utility which scans the system for available Mellanox devices (only mst PCI devices) and performs the necessary firmware updates. For further information on firmware update, please refer to Appendix C: “Booting HCA Device in Livefish Mode,” on page 173.

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.

2.2.1 mlxfwmanager Synopsis

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

where:

`-d|--dev DeviceName`	Perform operation for specified mst device(s). Run 'mst status' command to list the available devices. Multiple devices can be specified delimited by semicolons. A device list containing semicolons must be quoted.
`-h|--help`	Show this message and exit
`-v|--version`	Show the executable version and exit
`--query`	Query device(s) info
`--query-format Format`	(Query|Online query) output format, XML|Text - default Text
`-u|--update`	Update firmware image(s) on the device(s)
`-i|--image-file FileName`	Specified image file to use
`-D|--image-dir DirectoryName`	Specified directory instead of default to locate image files
`-f|--force`	Force image update
`-y|--yes`	Answer is yes in prompts
`--no`	Answer is no in prompts

`--clear-semaphore`	Force clear the flash semaphore on the device, No command is allowed when this flag is used. NOTE: May result in system instability or flash corruption if the device or another application is currently using the flash. Exercise caution.
`--exe-rel-path`	Use paths relative to the location of the executable
`-l|--list-content`	List file/Directory content, used with --image-dir and --image-file flags
`--archive-names`	Display archive names in listing
`--nofs`	Burn image in a non failsafe manner
`--log`	Create log file
`-L|--log-file LogFileName`	Use specified log file
`--no_fw_ctrl`	Do not use firmware Ctrl update
`--no-progress`	Do not show progress
`-o|--outfile OutputFileName`	Write to specified output file
`--online`	Fetch required FW images online from Mellanox 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

2.2.2 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:

a. 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.
```

b. 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.
```

c. Query XML

```
mlxfwmanager --query --query-format XML
<Devices>
    <Device pciName="/dev/mst/mt4099_pci_cr0" type="ConnectX3" psid="MT_1200111023" part-
Number="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" part-
Number="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>
```

2.2.3 Updating the Device

- To update a device on the machine, use the following command line[1]:

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

1. If only PXE rom needs update, please add -f to the command line

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:        0002c9000002
  Versions:         Current        Available
     FW             2.31.5050      2.32.5000
  Status:           Update required
---------
Found 1 device(s) requiring firmware update.
```

2.2.4 Updating the Device Online

To update the device online on the machine from Mellanox site, 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.
```

2.2.4.1 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_5_5_native
esxi_5_5_vmk
esxi_6_native
esxi_6_vmk
fbsd10_64
linux
linux_arm64
linux_ppc64
linux_ppc64le
linux_x64
windows
windows_x64

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

Example:

a. 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 - esxi_5_5_vmk/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 - esxi_6_vmk/mlxup
9 - linux/mlxup
```

```
10 - fbsd10_64/mlxup
11 - esxi_5_5_native/mlxup
12 - esxi_6_5_native/mlxup

<Release Notes>:
For more details, please refer to the following FW release notes:
    1- ConnectX3 (2.40.7000):    http://www.mellanox.com/pdf/firmware/ConnectX3-
FW-2_40_7000-release_notes.pdf
    2- ConnectX3Pro (2.40.7000): http://www.mellanox.com/pdf/firmware/ConnectX3-
Pro-FW-2_40_7000-release_notes.pdf
    3- Connect-IB (10.16.1020):  http://www.mellanox.com/pdf/firmware/ConnectIB-
FW-10_16_1020-release_notes.pdf
    4- ConnectX4 (12.18.2000):   http://www.mellanox.com/pdf/firmware/ConnectX4-
FW-12_18_2000-release_notes.pdf
    5- ConnectX4Lx (14.18.2000): http://www.mellanox.com/pdf/firmware/ConnectX4Lx-
FW-14_18_2000-release_notes.pdf
    6- ConnectX5 (16.19.1200):   http://www.mellanox.com/pdf/firmware/ConnectX5-
FW-16_19_1200-release_notes.pdf


Perform Download? [y/N]: y
Please wait while downloading Files to : '/tmp/DownloadDir'
        0 - linux_x64/mlxup : Done
        1 - esxi_5_5_vmk/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 - esxi_6_vmk/mlxup : Done
        9 - linux/mlxup : Done
        10 - fbsd10_64/mlxup : Done
        11 - esxi_5_5_native/mlxup : Done
        12 - esxi_6_5_native/mlxup : Done

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

b. Downloading firmware images/firmware update packages using custom key.

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

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


All :
-----

<Files>:
0 - Mellanox_Firmware_20170425.mfa
1 - linux_x64/mlxup
2 - esxi_5_5_vmk/mlxup
3 - windows/mlxup.exe
4 - esxi_6_native/mlxup
5 - windows_x64/mlxup.exe
6 - linux_ppc64/mlxup
7 - linux_arm64/mlxup
8 - linux_ppc64le/mlxup
9 - esxi_6_vmk/mlxup
10 - linux/mlxup
11 - fbsd10_64/mlxup
12 - esxi_5_5_native/mlxup
13 - esxi_6_5_native/mlxup

<Release Notes>:
For more details, please refer to the following FW release notes:
    1- ConnectX3 (2.40.7000):    http://www.mellanox.com/pdf/firmware/ConnectX3-FW-2_40_7000-release_notes.pdf
    2- ConnectX3Pro (2.40.7000): http://www.mellanox.com/pdf/firmware/ConnectX3Pro-FW-2_40_7000-release_notes.pdf
    3- Connect-IB (10.16.1020):  http://www.mellanox.com/pdf/firmware/ConnectIB-FW-10_16_1020-release_notes.pdf
    4- ConnectX4 (12.18.2000):   http://www.mellanox.com/pdf/firmware/ConnectX4-FW-12_18_2000-release_notes.pdf
    5- ConnectX4Lx (14.18.2000): http://www.mellanox.com/pdf/firmware/ConnectX4Lx-FW-14_18_2000-release_notes.pdf
    6- ConnectX5 (16.19.1200):   http://www.mellanox.com/pdf/firmware/ConnectX5-FW-16_19_1200-release_notes.pdf


Perform Download? [y/N]: y
Please wait while downloading Files to : '/tmp/DownloadDir'
        0 - Mellanox_Firmware_20170425.mfa : Done
        1 - linux_x64/mlxup : Done
        2 - esxi_5_5_vmk/mlxup : Done
        3 - windows/mlxup.exe : Done
        4 - esxi_6_native/mlxup : Done
        5 - windows_x64/mlxup.exe : Done
        6 - linux_ppc64/mlxup : Done
        7 - linux_arm64/mlxup : Done
        8 - linux_ppc64le/mlxup : Done
        9 - esxi_6_vmk/mlxup : Done
        10 - linux/mlxup : Done
        11 - fbsd10_64/mlxup : Done
        12 - esxi_5_5_native/mlxup : Done
        13 - esxi_6_5_native/mlxup : Done

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

2.2.5 Update Package for Mellanox Firmware

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 Mellanox firmware binary images, and the mlxfwmanager firmware update tool.

Update Package for Mellanox Firmware 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 Mellanox devices firmware versions
- Performs firmware updates if needed
- Cleans up temporary files

2.2.6 Update Package for Mellanox Firmware 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.

2.2.6.1 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:

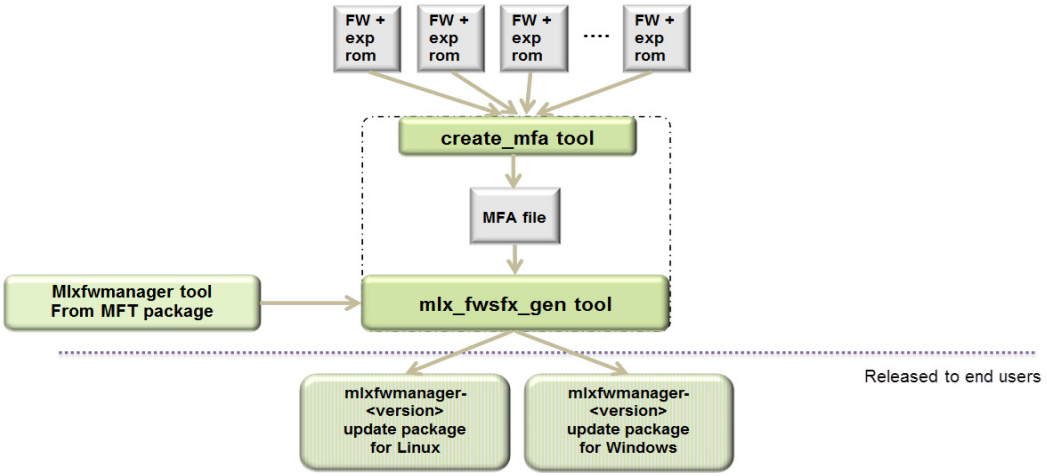
```
# mlx_fwsfx_gen --source-dir <FW images directory>  --out-dir <output directory>  [--sfx-name
<sfx file name>] [--phy-support --phy-img <phy-img>][--extra-args <args>]
```

where:

`--source-dir`	Directory containing Mellanox 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]

Figure 2: UPMF Package Generation Flow



2.2.6.2 UPMF Generation Example

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

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

Package name: /tmp/mlxfwmanager-20171004-1
Contents:
Source dirs: /tmp/fw-ConnectX-3Pro-dir
Adding file: /etc/mft/ca-bundle.crt
sfx_stub file: /usr/bin/mlx_sfx_stub
Creating intermediate MFA archive from binary files:
4779-314A-X00_Ax~ATT1090111023.bin
Huawei_TD70VMTA_VA_CX3Pro_2P_40G_Ax~HUA0020010017.bin
Inventec_U50_CX3Pro_10GE_A1~INV0010110023.bin
MCX342A-XCQ_Ax~MT_1680116023.bin
MCX353A-FCC_Ax~MT_1100111019.bin
mfa tool: /usr/bin/mlx_mfa_gen
mfa cmd: /usr/bin/mlx_mfa_gen -p /tmp/OMs1D5PvHq/srcs.mfa  -s /tmp/fw-ConnectX-3Pro-dir
```

```
Adding bins from /tmp/fw-ConnectX-3Pro-dir
Files copied: 5
Querying images ...
Files queried: 5
Compressing ... (this may take a minute)
Archive: /tmp/OMs1D5PvHq/srcs.mfa
Total time: 0m3s
Adding file: /tmp/OMs1D5PvHq/srcs.mfa
Adding file: /usr/bin/mlxfwmanager
Creating zip /tmp/OMs1D5PvHq/zippackage.zip
adding: srcs.mfa (deflated 0%)
adding: mlxfwmanager (deflated 52%)
adding: ca-bundle.crt (deflated 45%)
sfx auto-run command:
mlxfwmanager -u --log-on-update --ssl-certificate %ca-bundle.crt% %current-dir%  %argv%
Log name: /tmp/mlxfwmanager-20171004-1.log
```

2.2.6.3 UPMF Generation with PHY Binary Example

The below example packs 3 firmware binaries (named fw-ConnectX-3-1.bin, fw-ConnectX-3-2.bin, fw-ConnectX-3-3.bin) located in the directory `'/tmp/fw-ConnectX-3-dir/'` and a PHY image `'/tmp/Firmware_1.37.10_N32722.cld'` into a Linux UPMF package named `/tmp/mlxfwmanager-20141126-2.`

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

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

Files copied: 3

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

```
Archive: /tmp/YaH5BAoQ8q/srcs.mfa
Total time: 0m1s
Adding file: /tmp/YaH5BAoQ8q/srcs.mfa
Adding file: /usr/bin/mlxfwmanager
Copying /tmp/Firmware_1.37.10_N32722.cld to /tmp/C04TldeQHr/phy_mfa
Adding file: /tmp/Firmware_1.37.10_N32722.cld
Adding file: /usr/bin/mlxphyburn
Creating zip /tmp/YaH5BAoQ8q/zippackage.zip
  adding: srcs.mfa (deflated 0%)
  adding: ca-bundle.crt (deflated 45%)
  adding: phy_mfa/ (stored 0%)
  adding: phy_mfa/Firmware_1.37.10_N32722.cld (deflated 44%)
  adding: mlxfwmanager (deflated 57%)
  adding: mlxphyburn (deflated 60%)

sfx auto-run command:
 mlxfwmanager -u --log-on-update --ssl-certificate %ca-bundle.crt% %current-dir% %argv%

mlxphyburn auto-run command:
 mlxphyburn %device% -i ./phy_mfa/Firmware_1.37.10_N32722.cld b

Log name: /tmp/mlxfwmanager-20141126-2.log
```

2.2.7 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 Section 2.6, “mlxfwreset - Loading Firmware on 5th Generation Devices Tool”, on page 109.

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
```

2.3 mlxconfig - Changing Device Configuration Tool

The mlxconfig tool allows the user to change some of the device configurations without re-burning 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.

2.3.1 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: ConnectX®-3/ConnectX®-3 Pro, Connect-IB®, ConnectX®-4/ ConnectX®-4 Lx and ConnectX®-5/ConnectX®-5 Ex
- Changing device configurations enabled.

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

2.3.2 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`[1]	Show default and current configurations
`--force`[2]	Enables the user to skip some advanced checks performed by the tool
`-v|--version`	Displays version info
`-h|--help`	Displays help message
`-p|--private_key`	pem file for private key
`-u|--key_uuid`	keypair uuid

q[uery][3]	Queries the supported configurations
s[et]	Sets configurations to a specific device
set_raw	Sets raw configuration file (5th generation/Group II devices only)
i[show_confs]	Display informations 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 a Bin file from an XML file. XML input file name and bin output file name must be specified.
create_conf	Generate a Configuration file from an XML file. XML input file name and bin output file name must be specified.
apply	Apply a Configuration file. bin input file name must be specified.

1. 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.
2. The use of the --force flag is not recommended and is intended for advanced users only.
3. Query command will query a single device if a device is specified. Otherwise, it will query all devices on the machine

2.3.3 Examples of mlxconfig Usage

2.3.3.1 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_DESCIPTORS[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.
```

2.3.3.2 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_pcionf0

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.
```

2.3.3.3 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)
```

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

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

Example:

```
# mlxconfig -d pciconf-000:03:00.0 q
Device 1:
----------

Device type:   ConnectX-3
PCI Device:    pciconf-000:03:00.0

Configurations:              Next Boot
        SRIOV_EN             True(1)
        NUM_OF_VFS             16
        WOL_MAGIC_EN_P1      False(0)
        WOL_MAGIC_EN_P2      False(0)
```

2.3.5 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/mt4103_pci_cr0 set LINK_TYPE_P1=1 LINK_TYPE_P2=1

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

```
Device type:    ConnectX3Pro
PCI device:     /dev/mst/mt4103_pci_cr0

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.
```

2.3.6 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.
```

2.3.7 Using mlxconfig to Set preboot Settings

For a full description of the preboot configurable parameters refer to Table 7, “Supported Configurations and their Parameters,” on page 58 under "Preboot Settings".

Example: Enable boot option ROM on port 1, set boot retries to 3 and set the boot protocol to PXE.

```
# 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/mt4103_pciconf0 set BOOT_VLAN_P2=3

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

Device type:    ConnectX3Pro
PCI device:     /dev/mst/mt4103_pciconf0

Configurations:                    Next Boot       New
BOOT_VLAN_P2                        1              3

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

2.3.8 mlxconfig Raw Configuration Files

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

Set the raw configuration file:

```
# mlxconfig -f ./tlv_file.conf -d /dev/mst/mt4115_pciconf1 set_raw
Raw TLV #1 Info:
Length: 0xc
Version: 0
OverrideEn: 1
Type: 0x00000080
Data: 0xa0000000 0xa0020010 0x00000000
```

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

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

Raw TLV #4 Info:
Length: 0x4
Version: 0
OverrideEn: 1
Type: 0x00000082
Data: 0x0000000c

 Operation intended for advanced users.
 Are you sure you want to apply raw TLV file? ? (y/n) [n] : y
Applying... Done!
-I- Please reboot machine to load new configurations.
```

Never apply files from an unreliable source.

2.3.9 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.
```

2.3.10 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_conf0
```

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>
```

2.3.11 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 Section 2.3.10, 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
```

2.3.12 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 con-

figurations. To generate an XML file and fill it with the desired values, run the commands from Section 2.3.9, 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
```

2.3.13 mlxconfig create_conf Command

The `create_conf` command assists in creating an NV configuration file that can be used for Life-Cycle 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 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 and MLNX ID which are used for Secure Firmware Updates. Additionally, it supports two RSA keys of 2048 or 4096 bits length.

The NV configurations files must have the applicable_to configuration.

Example:

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

2.3.14 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!
```

2.3.15 Supported Configurations and their Parameters

Table 7: Supported Configurations and their Parameters (Sheet 1 of 18)

Feature	Parameter	Values
PCI Settings: Configure the device PCI settings	SRIOV_EN Enables or disables virtualization	False: Disable True: Enable
	NUM_OF_VFS=<NUM> Sets the number of virtual functions to allocate	1 to 127 (maximal number of virtual function may be smaller as it depends on the PCI BAR size and available system resources)
	[ConnectX-3 and ConnectX-3 Pro] LOG_BAR_SIZE Log (base 2) of the number of megabytes to be allocated per physical and virtual function	0 to 9 (maximal size may be smaller as it depends on the SR- IOV settings)
	NUM_PF_MSIX Number of MSI-X vectors and EQs per PF (5th generation/Group II)	0 to 127 (maximal size may be smaller/larger depending on the firmware)
	NUM_VF_MSIX Number of MSI-X vectors and EQs per VF (5th generation/Group II)	0 to 30 (maximal size may be smaller/larger depending on the firmware)
	PF_LOG_BAR_SIZE (for 5th generation/Group II devices) Log (base 2) of the size of a physical function's UAR BAR in megabytes	0 to 63 (maximal size may be smaller depending on the firmware)
	5th Generation Parameter: VF_LOG_BAR_SIZE Log (base 2) of the size of a virtual function's UAR BAR in megabytes	0 to 63 (maximal size may be smaller depending on the firmware)
PCI Settings: Configure the device PCI settings	**5th Generation Parameter**: NON_PREFETCHABLE_PF_BAR When set, the PF BAR 'prefetchable' bit is cleared. **Note:** PCI switches and operation systems have dedicated quotas for non-prefetchable memory hence, you may need to decrease log_pf_uar_bar_size to enable this feature.	False: Disable True: Enable
IB Dynamically Connect[1]**:** Configure the number of allocated local DC resources (DCR)	DCR_LIFO_SIZE The amount of total DCRs available to join linked-lists after hash DCRs	0 to 16777215 (maximal size may be smaller depending on the firmware)
	LOG_DCR_HASH_TABLE_SIZE Log2 of the hash table size minus 1	0 to 31 (maximal size may be smaller depending on the firmware)

Table 7: Supported Configurations and their Parameters (Sheet 2 of 18)

Feature	Parameter	Values
InfiniBand Boot Settings: Sets the partition key to be used by PXE boot	**4th Generation Parameters:** • BOOT_PKEY_P1 • BOOT_PKEY_P2 5th Generation Parameters: • BOOT_PKEY Partition key to be used by PXE boot	0-65535 0: Default
Internal Settings[1]: Configures internal settings of the HCA	INT_LOG_MAX_PAYLOAD_SIZE log2 of the maximal burst length when accessing PCI translation table	_4KB: 4KB burst length AUTOMATIC: Default
Preboot Boot Settings: Settings that control the legacy option ROM	**4th Generation Parameters:** • BOOT_OPTION_ROM_EN_P1 • BOOT_OPTION_ROM_EN_P2 5th Generation Parameters: • BOOT_OPTION_ROM_EN Disable/Enable boot option ROM	False: Disable True: Enable
	4th Generation Parameters: • BOOT_VLAN_EN_P1 • BOOT_VLAN_EN_P2 **5th Generation Parameters:** • BOOT_VLAN_EN Disable/Enable VLAN mode for network boot	False: Disable True: Enable
	BOOT_RETRY_CNT_P1 BOOT_RETRY_CNT_P2 Number of retries to attempt in case of boot failure from port1/2.	0 to7 7: infinite retries
	LEGACY_BOOT_PROTOCOL_P1 LEGACY_BOOT_PROTOCOL_P2 Boot protocol for port1/2	None: None - disable legacy boot PXE: PXE (DHCP/TFTP boot) iSCSI: iSCSI Both: PXE + iSCSI

Table 7: Supported Configurations and their Parameters (Sheet 3 of 18)

Feature	Parameter	Values
Preboot Boot Settings: Settings that control the legacy option ROM	**4th Generation Parameters:** • BOOT_VLAN_P1 • BOOT_VLAN_P2 **5th Generation Parameters:** • BOOT_VLAN VLAN ID for the network boot	0 to 4095
	LEGACY_BOOT_PROTOCOL Select boot protocol for legacy BIOS expansion ROM (FlexBoot)	0x0: NONE 0x1: PXE 0x2: ISCSI 0x3: BOTH - PXE + iSCSI
	BOOT_RETRY_CNT Number of retries to attempt in case of boot failure (FlexBoot)	0x0: NO_RETRIES 0x1: 1_RETRY 0x2: 2_RETRIES 0x3: 3_RETRIES 0x4: 4_RETRIES 0x5: 5_RETRIES 0x6: 6_RETRIES 0x7: UNLIMITED
	BOOT_LACP_DIS When set, FlexBoot does not respond to LACP configuration frames from the switch.	0: False 1: True
	4th Generation Parameters: • IP_VER_P1 • IP_VER_P2 **5th Generation Parameters:** • IP_VER **Note**: These parameters are relevant only for servers using legacy BIOS PXE boot (FlexBoot).	IPv4 IPv6 IPv4_IPv6 IPv6_IPv4 **Note**: If both values are configured, FlexBoot will try to boot with the second protocol only if DHCP parameters for the first protocol are not available, or if booting with the first protocol has failed.

Table 7: Supported Configurations and their Parameters (Sheet 4 of 18)

Feature	Parameter	Values
RoCE Congestion Control ECN[1]: RoCE Congestion Control ECN parameters	CLAMP_TGT_RATE_AFTER_-TIME_INC_P1 CLAMP_TGT_RATE_AFTER_-TIME_INC_P2 When receiving a CNP, the target rate should be updated if the transmission rate was increased due to the timer, and not only due to the byte counter	False: Disable True: Enable
	CLAMP_TGT_RATE_P1 CLAMP_TGT_RATE_P2 If set, whenever a CNP is processed, the target rate is updated to be the current rate	False: Disable True: Enable
	CNP_DSCP_P1 CNP_DSCP_P2 The DiffServ Code Point of the generated CNP for port1/2	0 to 63 0: Default
	CNP_802P_PRIO_P1 CNP_802P_PRIO_P2 The 802.1p priority value of the generated CNP for port1/2	0 to 7 7: Default
	DCE_TCP_G_P1 DCE_TCP_G_P2 Used to update the congestion estimator (alpha) once every DCE_TCP_RTT microseconds for the selected port	0 to 1023 64: Default

Table 7: Supported Configurations and their Parameters (Sheet 5 of 18)

Feature	Parameter	Values
RoCE Congestion Control ECN[1]: RoCE Congestion Control ECN parameters	DCE_TCP_RTT_P1 DCE_TCP_RTT_P2 The time between updates of the alpha value per port, in microseconds	0 to 131071 2: Default
	INITIAL_ALPHA_VALUE_P1 INITIAL_ALPHA_VALUE_P2 The initial value of alpha to use when receiving the first CNP for a flow. Expressed in a fixed point fraction of 2^10	0 to 1023
	MIN_TIME_BETWEEN_CNPS_P1 MIN_TIME_BETWEEN_CNPS_P2 Minimum time between sending cnps from the port, in microseconds	0 to131071 0: Default
	RATE_TO_SET_ON_FIRST_CNP_P1 RATE_TO_SET_ON_FIRST_CNP_P2 The rate that is set for the flow when a rate limiter is allocated to it upon first CNP received, in Mbps	0 to 131071 0: Default
	RATE_REDUCE_MONITOR_PERIOD_P1 RATE_REDUCE_MONITOR_PERIOD_P2 The minimum time between 2 consecutive rate reductions for a single flow. Rate reduction will occur only if a CNP is received during the relevant time interval	0 to 131071 2: Default
	RPG_AI_RATE_P1 RPG_AI_RATE_P2 The rate, in Mbits per second, used to increase rpTargetRate in the RPR_ACTIVE_INCREASE state	0 to 131071 10: Default
	RPG_BYTE_RESET_P1, RPG_BYTE_RESET_P2 Transmitted data between rate increases if no CNPs are received. Given in Bytes	0 to 32767 0: Disabled 150: Default
	CNP_RES_PRIO_MODE_P1/ CNP_RES_PRIO_MODE_P2 If set, CNP packets for this port contain priority from received request, if unset, use value from cnp_802p_prio	True False

Table 7: Supported Configurations and their Parameters (Sheet 6 of 18)

Feature	Parameter	Values
RoCE Congestion Control ECN[1]: RoCE Congestion Control ECN parameters	RPG_GD_P1, RPG_GD_P2 If a CNP is received, the flow rate is reduced at the beginning of the next rate_reduce_monitor_period interval to, (1-Alpha/Gd)*CurrentRate. RPG_GD is given as log2(Gd), where Gd may only be powers of 2	0 to 15 7: Default
	RPG_HAI_RATE_P1, RPG_HAI_RATE_P2 The rate, in Mbits per second, used to increase rpTargetRate in the RPR_HYPER_INCREASE state	0 to 131071 150: Default
	RPG_MAX_RATE_P1, RPG_MAX_RATE_P2 The maximum rate, in Mbits per second, at which an RP can transmit. Once this limit is reached, the RP rate limited is released and the flow is not rate limited any more	32 bit integer 0: Default (full port speed)
	RPG_MIN_DEC_FAC_P1, RPG_MIN_DEC_FAC_P2 The minimum factor by which the current transmit rate can be changed when processing a CNP. Value is given as a percentage	1 to 100 50: Default
	RPG_MIN_RATE_P1, RPG_MIN_RATE_P2 The minimum value, in Mb per second, for rate to limit	0 to 131071 2000: Default
	RPG_THRESHOLD_P1, RPG_THRESHOLD_P2 The number of times rpByteStage or rpTimeStage can count before the RP rate control state machine advances states	0 to 31 5: Default
	RPG_TIME_RESET_P1, RPG_TIME_RESET_P2 Time between rate increases if no CNPs are received. Given in u-seconds	0 to 131071 2: Default

Table 7: Supported Configurations and their Parameters (Sheet 7 of 18)

Feature	Parameter	Values
RoCE Congestion Control ECN[1]: RoCE Congestion Control ECN parameters	ROCE_CC_ALGORITHM_P1, ROCE_CC_ALGORITHM_P2 Congestion control algorithm for the specific port	ECN: ECN QCN: QCN
	ROCE_CC_PRIO_MASK_P1, ROCE_CC_PRIO_MASK_P2 Per priority enable disable bitmask for the specific port.	0 to 255 0: Default
VPI Settings: Configures the port's working mode	LINK_TYPE_P1 Configures port 1 working mode LINK_TYPE_P2 Configures port 2 working mode	IB: InfiniBand ETH: Ethernet VPI: VPI
	4th Generation Parameters: FORCE_MODE_P1 FORCE_MODE_P2	True False
	4th Generation Parameters: PHY_TYPE_P1 PHY_TYPE_P2	• 1: XAUI • 2: XFI • 3: SGMII
	4th Generation Parameters: XFI_MODE_P1 XFI_MODE_P2	0: _10G 1: _20G 2: _40G
Wake on LAN[2]: Enables/ disables Wake on LAN feature	WOL_MAGIC_EN_P1 (for ConnectX-3 and ConnecX-3 Pro) Enables or disables Wake on magic packet for port 1	False: Disable True: Enable
	WOL_MAGIC_EN_P2 (for ConnectX-3 and ConnecX-3 Pro) Enables or disables Wake on magic packet for port 2	False: Disable True: Enable
	WOL_MAGIC_EN[1] Enables or disables Wake on Magic packet for 5th Generation/Group II devices. Configuration is valid per Host and per Physical function.	False: disable True: Enable

Table 7: Supported Configurations and their Parameters (Sheet 8 of 18)

Feature	Parameter	Values
External Ports[1]: External physical port counters reading indicator	ALLOW_RD_COUNTERS Indicates whether reading external physical port counters by PPCNT is allowed. PORT_OWNER Indicates whether configuring/modifying external physical port parameters is allowed. RENEG_ON_CHANGE When cleared, a link-down/up sequence from the driver (PAOS) triggers re-negotiation of the link speed and parameters with the remote peer. When set, a link down/up sequence will trigger re-negotiation only if any link parameters were changed by a driver since last link negotiation. This parameter is applicable when the port_owner parameter is set for this PF	False: Disable True: Enable
LLDP Client Setting[1]	LLDP_NB_DCBX_P1 Enable DCBX for the first port (applicable when LLDP_NB_TX_MODE and LLDP_NB_RX_MODE are in ALL mode)	False: Disable True: Enable
	LLDP_NB_DCBX_P2 Enable DCBX for the second port (applicable when LLDP_NB_TX_MODE and LLDP_NB_RX_MODE are in ALL mode)	False: Disable True: Enable
	LLDP_NB_RX_MODE_P1 Enable the internal LLDP client, and define which TLV it will process.	• 0x0: OFF - Not listen for incoming LLDP BPDU (incoming LLDP frames will be routed to the host) • 0x1: MANDATORY - Listen for incoming LLDP frames, store only the mandatory LLDP BPDUs (1..3) • 0x2: ALL - Receive and store all incoming LLDP BPDUs

Table 7: Supported Configurations and their Parameters (Sheet 9 of 18)

Feature	Parameter	Values
LLDP Client Setting[1]	LLDP_NB_RX_MODE_P2 Enable the internal LLDP client, and define which TLV it will process.	• 0x0: OFF - Not listen for incoming LLDP BPDU (incoming LLDP frames will be routed to the host) • 0x1: MANDATORY - Listen for incoming LLDP frames, store only the mandatory LLDP BPDUs (1..3) • 0x2: ALL - Receive and store all incoming LLDP BPDUs
	LLDP_NB_TX_MODE_P1 Select which LLDP TLV will be generated by the NIC	• 0x0: OFF – NIC internal LLDP client will not send LLDP frames • 0x1: MANDATORY – Transmit only mandatory LLDP BPDU (ChassisID, PortID & TTL) • 0x2: ALL - Transmit optional LLDP BPDU if configured
	LLDP_NB_TX_MODE_P2 Select which LLDP TLV will be generated by the NIC	• 0x0: OFF – NIC internal LLDP client will not send LLDP frames • 0x1: MANDATORY – Transmit only mandatory LLDP BPDU (ChassisID, PortID & TTL) • 0x2: ALL - Transmit optional LLDP BPDU if configured

Table 7: Supported Configurations and their Parameters (Sheet 10 of 18)

Feature	Parameter	Values
QoS Configuration[1]	NUM_OF_TC_P1 Number of traffic classes, when DCB-X is enabled, this is the maximum number of TC that can negotiated with the remote peer.	8_TCS: 0x0 1_TC: 0x1 2_TCS: 0x2 3_TCS: 0x3 4_TCS: 0x4 5_TCS: 0x5 6_TCS: 0x6 7_TCS: 0x7
	NUM_OF_TC_P2 Number of traffic classes, when DCB-X is enabled, this is the maximum number of TC that can negotiated with the remote peer.	8_TCS: 0x0 1_TC: 0x1 2_TCS: 0x2 3_TCS: 0x3 4_TCS: 0x4 5_TCS: 0x5 6_TCS: 0x6 7_TCS: 0x7
	NUM_OF_VL_P1 Number of InfiniBand Virtual Lanes for this port.	15_VLS: 0x5 1_VL: 0x1 2_VLS: 0x2 4_VLS: 0x3 8_VLS: 0x4
	NUM_OF_VL_P2 Number of InfiniBand Virtual Lanes for this port.	15_VLS: 0x5 1_VL: 0x1 2_VLS: 0x2 4_VLS: 0x3 8_VLS: 0x4
LLDP NB DCBX[1]	DCBX_CEE_P1 Enable DCBX in CEE mode.	False: Disable True: Enable
	DCBX_CEE_P2 Enable DCBX in CEE mode.	False: Disable True: Enable
	DCBX_IEEE_P1 Enable DCBX in IEEE mode.	False: Disable True: Enable
	DCBX_IEEE_P2 Enable DCBX in IEEE mode.	False: Disable True: Enable
	DCBX_WILLING_P1 Allow the NIC to accept DCBX configuration from the remote peer.	False: Disable True: Enable
	DCBX_WILLING_P2 Allow the NIC to accept DCBX configuration from the remote peer.	False: Disable True: Enable

Table 7: Supported Configurations and their Parameters (Sheet 11 of 18)

Feature	Parameter	Values
Keep Link Up[1]	KEEP_ETH_LINK_UP_P1 / KEEP-_ETH_LINK_UP_P2 When set, the NIC keeps the link up as long as the server is not in standby mode (Ethernet only).	False True
	KEEP_IB_LINK_UP_P1 / KEEP-_IB_LINK_UP_P2 When set, the NIC keeps the link up as long as the server is not in standby mode (InfiniBand only).	False True
	KEEP_LINK_UP_ON_BOOT_P1 / KEEP-_LINK_UP_ON_BOOT_P2 When set, the NIC keeps the link up as long as the server is not in standby mode and a driver is not initialized.	False True
	KEEP_LINK_UP_ON_STANDBY_P1 / KEEP_LINK_UP_ON_STANDBY_P2 When set, the NIC keeps the link up from power-up until the server is turned on, and when the server is in standby mode.	False True
	DO_NOT_CLEAR_PORT_STATS_P1/ DO_NOT_CLEAR_PORT_STATS_P2 When set, the port statistic counters are not cleared on first host init. When cleared the port statistics are cleared on first host init	False True
MPFS CONF	DUP_MAC_ACTION_P1 / DUP_MAC_AC-TION_P2 Defines the forwarding behavior in MPFS for MACs which are duplicated in more than one PF.	`LAST_CFG`: The last PF which added the MAC will receive the traffic LOAD_BALANCE: MPFS will load-balance IPv4 traffic for packets with destination MAC addresses that belong to more than one PF between all PFs that have this MAC

Table 7: Supported Configurations and their Parameters (Sheet 12 of 18)

Feature	Parameter	Values
MPFS CONF	IB_ROUTING_MODE_P1 IB_ROUTING_MODE_P2 Selects the routing mode for InfiniBand among the PFs (and hosts, if applicable): • GID - each PF has a unique GID • LID - each PF has a unique LID	0x0: GID 0x1: LID
	SRIOV_IB_ROUTING_MODE_P1 SRIOV_IB_ROUTING_MODE_P2 Selects the IB routing mode for Virtual Functions: • GID - each PF has a unique GID • LID - each PF has a unique LID	0x0: GID 0x1: LID
SW OFFLOAD CONF[1]	CQE_COMPRESSION Configures which algorithm should be used by the NIC in order to decide when to activate CQE compression based on PCIe bus condition.Note that the driver can enable compression on a per CQE basis.	AGGRESSIVE BALANCED
	LRO_LOG_TIMEOUT0/LRO_LOG_TIMEOUT1/LRO_LOG_TIMEOUT2/LRO_LOG_-TIMEOUT3 Log2 of Large Receive Offload (LRO) timeout #0/1/2/3, in microseconds. Driver can select one of the 4 configured LRO timeouts on a per Qp basis in run-time (lro_timeout_period_usecs field of the TIR context).	Numerical value
	IP_OVER_VXLAN_EN	0: False 1: True
	IP_OVER_VXLAN_PORT	0 to 65535

Table 7: Supported Configurations and their Parameters (Sheet 13 of 18)

Feature	Parameter	Values
ROM DEBUG LEVEL[1]	BOOT_DBG_LOG When this bit is set, FlexBoot will generate debug log at the levels specified by the following parameters. Logs are dumped to the display and to the serial console (if configured).	True False
	BOOT_DBG_LOG_ARP Debug level for FlexBoot's Address Resolution Protocol (ARP)	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_DHCP Debug level for FlexBoot's DHCP	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_DHCP6 Debug level for FlexBoot's DHCPv6	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_DRIVER_SETTINGS Debug level for non volatile configuration management	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_DRV Debug level for FlexBoot's driver (init, tear-down, poll CQ etc)	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_IP Debug level for FlexBoot's IPv4 stack	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL

Table 7: Supported Configurations and their Parameters (Sheet 14 of 18)

Feature	Parameter	Values
ROM DEBUG LEVEL[1]	BOOT_DBG_LOG_IPV6 Debug level for FlexBoot's IPv6 stack	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_NDP Debug level for FlexBoot's IPv6 Neighbor Discovery Protocol	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_NDRV Debug level for FlexBoot's NODNIC driver (init, teardown, poll CQ etc)	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_NDRV_CMD Debug level for FlexBoot's NODNIC Driver interface	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_NDRV_DEV Debug level for FlexBoot's NODNIC Driver tear-down and bring ups operations	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_NDRV_PORT Debug level for FlexBoot's NODNIC Driver port management	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_NETDEVICE Network Device Link status and traffic	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_PXE_UNDI Debug level for FlexBoot's UNDI interface	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_ROMPREFIX Debug level for FlexBoot's driver bootstrap code	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL

Table 7: Supported Configurations and their Parameters (Sheet 15 of 18)

Feature	Parameter	Values
ROM DEBUG LEVEL[1]	BOOT_DBG_LOG_STATUS Debug level for status update interface	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_STP Debug level for FlexBoot's Spanning Tree Protocol	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_TCP Debug level for FlexBoot's TCP stack	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_TCPIP Debug level for FlexBoot's TCP/IP stack.	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_TFTP Debug level for FlexBoot's TFTP stack.	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_UDP Debug level for FlexBoot's UDP stack.	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
	BOOT_DBG_LOG_URI Debug level for FlexBoot's URI parsing code	0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL
ROM UEFI DEBUG LEVEL[1]	UEFI_DEBUG_LOGS Select the output channel device logs generated by the NIC/HBA UEFI expansion ROM. Note: When DISABLED, all the parameters in this TLV should be hidden by mlxconfig.	0x0: DISABLED 0x1: STDOUT 0x2: STDERR
	UEFI_DEBUG_LOG_MIN_SEVERITY Set the minimal severity of logs that will be generated by NIC/HBA UEFI expansion ROM	0x0: ERROR 0x1: WARNING 0x2: INFO 0x3: DEBUG
	UEFI_DEBUG_LOG_LOAD Enable UEFI debug level logs for load events	False True

Table 7: Supported Configurations and their Parameters (Sheet 16 of 18)

Feature	Parameter	Values
ROM UEFI DEBUG LEVEL[1]	UEFI_DEBUG_LOG_BLKIO Enable UEFI debug level logs for BlkIo Driver. UEFI_DEBUG_LOG_BM Enable UEFI debug level logs for Boot Manager. UEFI_DEBUG_LOG_CACHE Enable UEFI debug level logs for Memory range cachebility changes. UEFI_DEBUG_LOG_DISPATCH Enable UEFI debug level logs for PEI/DXE/SMM Dispatchers. UEFI_DEBUG_LOG_EVENTS Enable UEFI debug level logs of Event. UEFI_DEBUG_LOG_FS Enable UEFI debug level logs for EFI file system accesses. UEFI_DEBUG_LOG_GCD Enable UEFI debug level logs for Global Coherency Data base changes. UEFI_DEBUG_LOG_INIT Enable UEFI debug level initialization log. UEFI_DEBUG_LOG_PAGE Enable UEFI debug level logs for page Alloc & Free. UEFI_DEBUG_LOG_POOL Enable UEFI debug level logs for pool Alloc & Free. UEFI_DEBUG_LOG_SNI Enable UEFI debug level logs SNI Driver. UEFI_DEBUG_LOG_UNDI Enable UEFI debug level logs UNDI Driver. UEFI_DEBUG_LOG_VERBOSE High verbosity UEFI expansion ROM debug log. Please note that high verbosity may impact boot performance.	True False

Table 7: Supported Configurations and their Parameters (Sheet 17 of 18)

Feature	Parameter	Values
POWER CONF[1]	ADVANCED_POWER_SETTINGS Enable manual override for the default power settings. When Set, additional power settings parameter in NV_POWER_CONF are valid and configurable. When cleared, additional power settings parameter in NV_POWER_-CONF are ignored.	True False
	DISABLE_SLOT_POWER_LIMITER When cleared, the device is not allowed to consume more than 25W from the PCIe power rails, unless a PCI slot power limit message which a new power limit is received. When set, the slot power limiter is disabled, and the device is allowed to consume more than 25W from the PCIe power rails. Note: if the power limiter is active and there is not enough power, the device will shut down the network modules.	True False
	SW_RECOVERY_ON_ERRORS	True False
	RESET_WITH_HOST_ON_ERRORS	True False
PCI CONF[1]	ADVANCED_PCI_SETTINGS Show advanced PCI settings.	True False
	FORCE_ETH_PCI_SUBCLASS Force the PCI function identify with Ethernet subclass (00h). Supported only when ADVANCED_PCI_SET-TINGS is set.	True False
	IBM_CAPI_EN Enable IBM's Coherent Accelerator Processor Interface (CAPI) mode. Supported only when ADVANCED_PCI_SETTINGS is set.	True False
	IBM_TUNNELED_ATOMIC_EN	True False
	EXP_ROM_PXE_ENABLE	True False
	EXP_ROM_UEFI_x86_ENABLE	True False
	EXP_ROM_UEFI_ARM_ENABLE	True False

Table 7: Supported Configurations and their Parameters (Sheet 18 of 18)

Feature	Parameter	Values
MEMIC Conf[1]	MEMIC_BAR_SIZE The amount of BAR size assigned for MEMIC. The size in bytes	Numerical value
	MEMIC_SIZE_LIMIT The maximum amount of internal device memory that can be consumed by the MEMIC application.	0x0: DISABLED 0x1: 256KB 0x2: 512KB 0x3: 1024KB
Global Conf	**[ConnectX-3/ConnectX-3 Pro]** CQ_TIMESTAMP When set, IEEE1588 (PTP) hardware time-stamping capability reporting (cq_timestamp) is enabled	1- True 0 - False
HCA Conf[1]	MULTI_PORT_VHCA_EN When set, the PF reports the multi-port-vHCA capability.	0: False 1: True
ROM UEFI Conf[1]	UEFI_HII_EN When set, UEFI HII menus are enabled for this PCI function.	0: False 1: True
Host Chaining Configuration	HOST_CHAINING_MODE	False True
	HOST_CHAINING_DESCRIPTORS	Array of parameters that take unsigned integer values. To set a value for the fifth parameter: HOST_CHAINING_-DESCRIPTORS[5]=8
	HOST_CHAINING_TOTAL_BUFFER_SIZE	Array of parameters that take unsigned integer values. To set a value for the fifth parameter: HOST_CHAINING_-TOTAL_BUFFER_-SIZE[5]=8

1. 5th Generation devices only.
2. Only for supported devices listed in this section.

The default value for the parameters listed in the table above is firmware dependent.

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 SRIOV, please refer to "Troubleshooting" on page 156.

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).

2.4 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.

2.4.1 flint Synopsis

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

2.4.1.1 Switches Options

`-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.
`-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,log2-size,num_of_flashes>`	Use the given parameters to access the flash instead of reading them from the flash. Supported parameters: • Type: The type of the flash, such as: M25PXxx, M25Pxx, SST25VFxx, W25QxxBV, W25Xxx, AT25DFxxx, S25FLXXXP • log2size: The log2 of the flash size • num_of_flashes: the number of the flashes connected to the device
`-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, repectively: node, port1, port2 and system image GUID. **NOTE:** port2 guid must be specified even for a single port HCA. The HCA ignores this value. It can be set to 0x0. Commands affected: burn, sg
`-h[elp]`	Prints this message and exits
`-hh`	Prints extended command help
`-i[mage] <image>`	Binary image file. Commands affected: burn, verify
`-log <log_file>`	Prints the burning status to the specified log file

-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, repectively. Commands affected: burn, sg **NOTE:** -mac/-macs flags are applicable only for Mellanox Technologies 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.
-nofs	Burns image in a non failsafe manner.
--allow_rom_change	Allows burning/removing a ROM to/from Firmware image when product version is present.
-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.
-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 Mellanox Methodologies. Commands affected: burn, sg
-use_image_guids	Burn (guids/uids/macs) as appears in the given image. Commands affected: burn
-use_image_ps	Burn vsd as appears in the given image - do not keep existing VSD on flash. Commands affected: burn
-use_image_rom	Do not save the ROM which exists in the device. Commands affected: burn
-use_dev_rom	Save the ROM which exists in the device (FS3 and FS4 image only). Commands affected: burn
--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
--no_fw_ctrl	Do not attempt to work with the firmware Ctrl update commands.
-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

`--use_fw`	Access to flash using FW (ConnectX3/ConnectX3Pro Device Only) Commands affected: all
`--private_key <key_file>`	Path to PEM formatted private key to be used by the sign command
`--key_uuid <uuid_file>`	UUID matching the given private key to be used by the sign command
`--private_key2 <key_file>`	Path to PEM formatted 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

1. The -mac and -macs options are applicable only to Mellanox Technologies Ethernet adapter and switch devices.
2. When accessing SwitchX via I2C or PCI, the -override_cache_replacement flag must be set.

2.4.1.2 Command Parameters

The **flint** utility commands are:

Common FW Update and Query

`b[urn]`	Burn flash
`q[uery] [full]`	Query misc. flash/firmware characteristics, use "full" to get more information.
`v[erify]`	Verify entire flash
`swreset`	SW reset the target un-managed switch device. This command is supported only in the In-Band access method.

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.
`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

`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).
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.

2.4.2 Burning a Firmware Image

The FLINT utility enables you to burn the Flash from a binary image.

To burn the entire Flash from a raw binary image, use the following command line:

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

where:

`device`	Device on which the flash is burned.

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

➢ ***To burn a firmware image:***

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

```
# flint -d /dev/mst/mt4099_pci_cr0 -i fw-4099-2_31_5050-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_31_5050-MCX354A-FCB_A2.bin -guid 1234567dead-
beef burn
```

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

```
# flint -d /dev/mst/mt4099_pci_cr0 -i fw-4099-2_31_5050-MCX354A-FCB_A2.bin -mac 1234567dead-
beef 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.

```
# flint -d /dev/mst/mt4099_pci_cr0 -i fw-4099-2_31_5050-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
```

2.4.3 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_pciconf0 query
```

- Query the FW image file.

```
# flint -i 25408-2_30_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.

2.4.4 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
```

2.4.5 Performing Checksum Calculation on Image/Device

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> checksum
```

➢ ***To perform a checksum on a firmware image, run the following command line:***

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

where:

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

Examples:

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

flint -d /dev/mst/mt4117_pciconf0 cs
-I- Calculating Checksum ...
Checksum: 68ddae6bfe42f87f09084f3f468a35c6
```

2.4.6 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 rrom example.mrom
# flint -d /dev/mst/mt4099_pci_cr0 q
    Image type:      FS2
    FW Version:      2.31.5050
    FW Release Date: 4.5.2014
    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
```

Example:

```
 # flint -d /dev/mst/mt4099_pci_cr0 drom
  Removing ROM image   - OK
  Restoring signature  - OK
```

2.4.7 Setting GUIDs and MACs

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

2.4.7.1 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_pciconf0 q
-W- Running quick query - Skipping full image integrity checks.

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:                                 f4521401b8a1     f4521401b8a2
VSD:
PSID:            MT_1090120019

# flint -d /dev/mst/mt4099_pciconf0 -guid 0x452140300abadaba -mac 0x300abadaba sg

-W- GUIDs are already set, re-burining image with the new GUIDs ...
You are about to change the Guids/Macs/Uids on the device:
                          New Values              Current Values
        Node  GUID:     452140300abadaba        f45214030001b8a0
        Port1 GUID:     452140300abadabb        f45214030001b8a1
        Port2 GUID:     452140300abadabc        f45214030001b8a2
        Sys.Image GUID: 452140300abadabd        f45214030001b8a3
        Port1 MAC:          00300abadaba            f4521401b8a1
        Port2 MAC:          00300abadabb            f4521401b8a2

 Do you want to continue ? (y/n) [n] : y
Burning FS2 FW image without signatures - OK
Restoring signature                     - OK

# flint -d /dev/mst/mt4099_pciconf0 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
```

2.4.7.2 5th Generation (Group II) Devices

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

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

1. Change GUIDs for device.

```
# flint -d /dev/mst/mt4113_pciconf0 q
Image type:      FS3
FW Version:      10.10.3000
FW Release Date: 29.4.2014
Description:     UID                GuidsNumber  Step
```

```
Base GUID1:      0002c903002ef500        8        1
Base GUID2:      0002c903002ef508        8        1
Base MAC1:       0002c92ef500            8        1
Base MAC2:       0002c92ef508            8        1
Image VSD:
Device VSD:      VSD
PSID:            MT_1240110019

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

-W- Firmware flash cache access is enabled. Running in this mode may cause the firmware to
hang.
Updating GUID section - OK
Updating ITOC section - OK
Restoring signature   - OK

# flint -d /dev/mst/mt4113_pciconf0 q
Image type:      FS3
FW Version:      10.10.3000
FW Release Date: 29.4.2014
Description:     UID                GuidsNumber  Step
Base GUID1:      00002c123456abcd        8        1
Orig Base GUID1: 0002c903002ef500        8        1
Base GUID2:      00002c123456abd5        8        1
Orig Base GUID2: 0002c903002ef508        8        1
Base MAC1:       00002c56abcd            8        1
Orig Base MAC1:  0002c92ef500            8        1
Base MAC2:       00002c56abd5            8        1
Orig Base MAC2:  0002c92ef508            8        1
Image VSD:
Device VSD:      VSD
PSID:            MT_1240110019
```

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

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

```
# flint -d /dev/mst/mt4113_pciconf0 q
Image type:      FS3
FW Version:      10.10.3000
FW Release Date: 29.4.2014
Description:     UID                GuidsNumber  Step
Base GUID1:      0002c903002ef500        8        1
```

```
Base GUID2:      0002c903002ef508        8        1
Base MAC1:       0002c92ef500            8        1
Base MAC2:       0002c92ef508            8        1
Image VSD:
Device VSD:      VSD
PSID:            MT_1240110019

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

-W- Firmware flash cache access is enabled. Running in this mode may cause the firmware to
hang.
Updating GUID section - OK
Updating ITOC section - OK
Restoring signature   - OK

# flint -d /dev/mst/mt4113_pciconf0 q
Image type:      FS3
FW Version:      10.10.3000
FW Release Date: 29.4.2014
Description:     UID                GuidsNumber  Step
Base GUID1:      0000000000000001        2        1
Orig Base GUID1: 0002c903002ef500        8        1
Base GUID2:      0000000000000003        2        1
Orig Base GUID2: 0002c903002ef508        8        1
Base MAC1:       000000000001            2        1
Orig Base MAC1:  0002c92ef500            8        1
Base MAC2:       000000000003            2        1
Orig Base MAC2:  0002c92ef508            8        1
Image VSD:
Device VSD:      VSD
PSID:            MT_1240110019
```

3. Change GUIDs for image.

```
# flint -i /tmp/connect-ib.bin q
Image type:      FS3
FW Version:      10.10.3000
FW Release Date: 29.4.2014
Description:     UID                GuidsNumber  Step
Base GUID1:      0002c903002ef500        8        1
Base GUID2:      0002c903002ef508        8        1
Base MAC1:       0002c92ef500            8        1
Base MAC2:       0002c92ef508            8        1
Image VSD:
Device VSD:      VSD
PSID:            MT_1240110019

# flint -i /tmp/connect-ib.bin -uid 000123456abcd sg
Updating GUID section - OK
Updating ITOC section - OK
Restoring signature   - OK

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

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

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

-W- Firmware flash cache access is enabled. Running in this mode may cause the firmware to
hang.
Updating GUID section - OK
Updating ITOC section - OK
Restoring signature   - OK

# flint -d /dev/mst/mt4115_pciconf0 q
Image type:      FS3
FW Version:      12.0100.5630
FW Release Date: 23.3.2015
```

```
Description:     UID               GuidsNumber
Base GUID:       e41d2d0300570fc0       4
Base MAC:        e41d2d570fc0           4
Image VSD:
Device VSD:
PSID:            MT_2190110032

add note:
GUIDs and MACs can be changed separately on ConnectX4
```

2.4.7.3 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.

2.4.7.3.1 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 -wrimage./fw-ConnectX3-rel.bin
-striped_image -blank_guids


-I- Generating image ...
-I- Image generation completed successfully.
```

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

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

Image type:      FS2
FW Version:      2.31.5050
FW Release Date: 4.5.2014
```

```
Device ID: 4099
Description:     Node             Port1            Port2            Sys image
GUIDs:           ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
MACs:                                 ffffffffffffff     ffffffffffffff

VSD:             n/a
PSID:            MT_1090120019

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

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

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

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

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

Image type:      FS2
FW Version:      2.31.5050
FW Release Date: 4.5.2014
Device ID: 4099
Description:      Node             Port1           Port2           Sys image
GUIDs:            0002c9030abcdef0 0002c9030abcdef1 0002c9030abcdef2 0002c9030abcdef3
MACs:                                 0002c9bcdef1 0002c9bcdef2
VSD:              n/a
PSID:             MT_1090120019
```

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

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

➢ ***To pre-burn an image with Specific GUIDs/MACs for Each Device:***

1. Generate the base image with arbitrary default GUIDs

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

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

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

3. (**Optional**) Query the image to verify the GUIDs are set. The "-striped_image" flag must be specified when querying a striped image.

```
# flint -i ./fw-ConnectX3-rel.bin -striped_image q
Image type: FS2
FW Version: 2.31.5050
FW Release Date: 4.5.2014
Device ID: 4099
Description:      Node             Port1           Port2           Sys image
GUIDs:            0002c9030abcdef0 0002c9030abcdef1 0002c9030abcdef2 0002c9030abcdef3
MACs:                                0002c9bcdef1 0002c9bcdef2
VSD:              n/a
PSID:             MT_1090120019
```

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

2.4.8 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
```

2.4.9 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.

2.4.9.1 For 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 Appendix 2.4.17,“Secure Host,” on page 97

2.4.9.2 For 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.
```

2.4.10 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
```

2.4.11 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_conf01 ww 0x10008 0x5a445a44
```

2.4.12 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.

2.4.13 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 sector 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
```

2.4.14 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
```

2.4.15 Firmware Time-stamping 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.

2.4.15.1 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.

2.4.15.2 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.

2.4.15.3 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.

2.4.15.4 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>
```

2.4.15.5 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.

2.4.15.6 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.

2.4.15.7 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.

2.4.16 Flint/mlxburn Limitations

When running flint/mlxburn via an MTUSB-1 device, a burn/query command may take up to 45 minutes to complete.

To accelerate the burn process add the flag `-no_flash_verify` to the command line which skips the flash verification step. This flag, however, does not verify if the image is burnt correctly.

Burning an image to a ConnectX®-3 adapter in Flash recovery mode may fail on some server types (that use PCIe spread spectrum). The tool may not be able to recognize the device's PCI CONF0 or the image burn may not complete successfully.

To burn the device, use the MTUSB-1 connection.

To load the newly burnt firmware image, a driver restart is required for ConnectX-3/ConnectX-3 Pro cards.

For fifth generation (Group II) devices, run the mlxfwreset tool or reboot the system.

2.4.17 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 Section 2.4.17.4, “Key Loss Recovery”, on page 99

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.

2.4.17.1 Using Secure Host

Secure Host feature is supported for all Mellanox 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 Section 2.4.17.1.1, “Generating/Burning a Firmware Supporting Secure Host”, on page 98).

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

2.4.17.1.1 Generating/Burning a Firmware Supporting Secure Host

1. Make sure you have INI and mlx files suitable for the device.

   Both files are available for download at:

   http://www.mellanox.com/page/custom_firmware_table

   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 -wrimage 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.

2.4.17.1.2 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.

2.4.17.2 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
```

2.4.17.3 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 -wrimage fw-4099.unse-
cure.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
```

2.4.17.4 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 Appendix D: “Burning a New Device,” on page 175)

4. Remove the flash-not-present jumper.
5. Reboot the machine
6. Re-set the hardware access key

2.4.18 Secure Firmware Update

Secure Firmware Update is only supported in ConnectX-4/ConnectX-4 Lx and ConnectX-5/ ConnectX-5 Ex adapter cards.

“Secure firmware updates” 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 updates” 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 utilize the same ‘fail safe’ upgrade procedures, so events like power failure during update should not leave the device in an unstable state.

2.4.18.1 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.

Example:

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

Example:

```
#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"
```

2.4.18.2 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
```

2.4.18.3 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
```

2.4.18.4 Secure Firmware Implications on Burning Tools

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

Flint query under secure mode:

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

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 Section 2.4.18.1, “Signing Binary Image Files”, on page 100.

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:          FS3
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
```

2.5 mlxburn - Firmware Image Generator and Burner

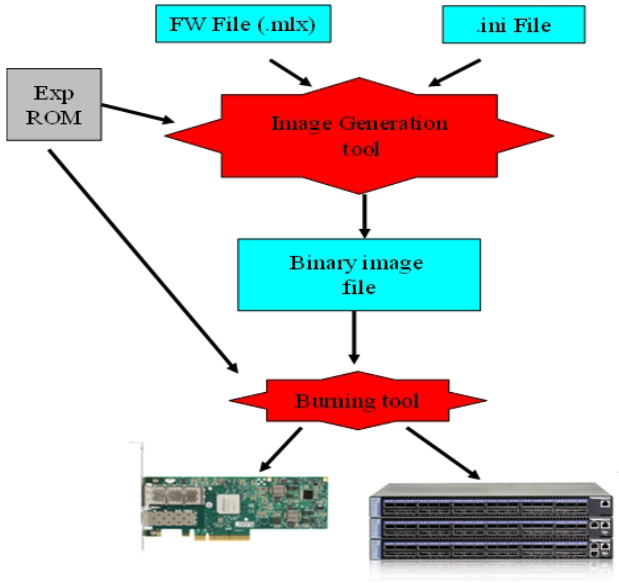
mlxburn is a tool for firmware (FW) image generation and/or for burning a firmware image to the Flash/EEPROM attached to a Mellanox device. Both functions or a single function of mlxburn can be activated by means of command line options (see Section 2.5.3, “mlxburn Synopsis”). It can also query for firmware attributes (e.g., firmware version, GUIDs, etc.) and VPD info of adapter cards and switch systems.

mlxburn allows for customization of standard Mellanox firmware for OEM specific needs (e.g., a specific adapter board type). See Section 2.5.2, “Customizing Firmware”, on page 103.

2.5.1 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 Mellanox 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/EERPROM attached to a Mellanox device in the second stage. The burning process retains device specific data such as GUIDs, UIDs, MACs, VSD, and BSN. Also, the burn process is failsafe by default.

Figure 3: FW Generation and Burning



mlxburn runs both stages by default, but it may perform only one by means of command options. If the '-wrimage' is specified (see Section 2.5.3, "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 '-wrimage' option).

2.5.2 Customizing Firmware

A Mellanox 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 '-wrimage <target file>'. To actually burn the image to the Flash/EEPROM attached to a Mellanox 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.

2.5.3 mlxburn Synopsis

```
#mlxburn [-h][-v] <-dev mst-device|-wrimage fw-image>
<-fw mellanox-fw-file|-image fw-image|-img_dir img_direcory|-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>]
[-fwver][1][-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>]
```

1. The "-fwver" flag is not supported in Connect-IB®, Switch-IB™, ConnectX-4® and ConnectX-5® devices.

where:

-dev_type <mellanox-device-number>	mlxburn must know the device type in order to work properly Use this flag if device type auto-detection fails. Example: -dev_type 23108 Supported Mellanox device types: • HCAs/NICs: 4099, 4103, 4113, 4115, 4117, 4119. • Switches: 51000, 52000, 52100, 53000.
-fw <mellanox-fw-file>	Specify Mellanox 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
-dev <mst-dev>	Burn the image using the given mst device
-exp_rom <exp-rom-file>	Integrate the given expansion rom file to the FW image. The given file may be in .img or bin/.rom (raw binary) format. • 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). **NOTE:** Exp rom auto detection is done for devices that are already burned with an exp-rom image. • If "-exp_rom AUTO" is specified for a device with no exp-rom, it would be burnt with no exp rom. To add exp-rom to a device, manually supply the exp rom file to use
-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 -wrimage 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	• When a device is given: Display current loaded firmware version. • When a FW file is given (-fw flag): Display the file FW version. **Note:** The "-fwver" flag is not supported in Connect-IB® devices.
-h	Display a short help text
-image <fw-image-file>	Do not generate image. Use the given fw image instead
-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
-query	Query the HCA/Switch device for firmware details, e.g. Firmware Version, GUIDs etc. In addition to the above flags, Mlxburn can also accept the following flags/options, which are passed to the underlying burning tool: -banks -use_image_ps -skip_is -mac(s) -guid(s) -sysguid -vsd -ndesc -bsn -pe_i2c -se_i2c -is3_i2c -no -uid(s) -log -blank_guids -flash_params -allow_psid_change -no_flash_verify -use_image_rom -override_cache_replacement -use_image_guids See the flint tool documentation for HCA/4th gen switches/Bridge burning options.
-v	Print version info and exit
-V <INFORM|WARNING|DEBUG>	Set verbosity level. Default is WARNING
-vpd[1,2]	Display the read only section of the PCI VPD (Vital Product Data) of the given device
-vpd_prog_rw<rw-keywords-file>[1,2]	(**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". 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 White spaces before and after VALUE are trimmed
-vpd_rw[1,2]	(**on Linux only**): Display also the read/write section of the PCI VPD of the given device.
-vpd_set_keyword <keyword-assignment>[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-keywords-file 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.

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

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.

2.5.3.1 Mellanox Connect-IB®, Switch-IB™, Switch-IB™ 2, 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 write-able 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).

2.5.3.2 Additional mlxburn Options

The following is a list of additional options. Please see Chapter 2.2, “mlxfwmanager - Firmware Update and Query Tool” for the HCA options.[1]

```
[-use_image_ps] [-skip_is] [-mac(s)] [-guid(s)] [-sysguid] [-vsd] [-uid(s)] [-log] [-
blank_guids] [-flash_params] [-allow_psid_change] [-no_flash_verify] [-use_image_rom] [-ocr] [-
use_image_guids] [-bank]
```

2.5.4 Examples of mlxburn Usage

2.5.4.1 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 -wrimage ./fw-4099.bin
```

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

- 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
```

2.5.4.2 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 -wrimage 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
```

2.5.4.3 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 -wrimage fw-ConnectX-4-MCX-456A-ECA_Ax.bin -base_guid 0x0002c903002ef500
```

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

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

2.5.4.4 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 -wrimage 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
```

2.5.4.5 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 -wrimage 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
```

2.5.4.6 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.

```
# mlxi2c -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
```

2.5.4.7 Spectrum™ Examples

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

```
mlxburn -fw FW/fw-SwitchEN.mlx -c FW/MSN2700-Cxxx_Ax.ini -wrimage 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
```

2.5.4.8 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 -wrimage 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
```

2.5.4.9 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 -wrimage 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
```

2.5.5 Exit Return Values

The following exit values are returned:

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

2.6 mlxfwreset - Loading Firmware on 5th Generation Devices Tool

mlxfwreset (Mellanox firmware reset) tool enables the user to load updated firmware without having to reboot the machine.

mlxfwreset supports 5th Generation (Group II) HCAs with ISFU[1] supported Firmware.

2.6.1 Tool Requirements

- Access to device through configuration cycles (pciconf)
- Firmware supporting ISFU
  - Connect-IB: v10.10.3000 or above
  - ConnectX-4: v12.0100.0000 or above
  - ConnectX-4 Lx: v14.0100.0000 or above
- Device's firmware updated with latest MFT burning tools (flint/mlxburn/mlxfwmanager)
- Supported devices: Connect-IB, ConnectX-4/ConnectX-4 Lx/ConnectX-5
- Supported OSs: FreeBSD10.02, FreeBSD11, Linux, Windows

Exact reset level needed to load new firmware may differ, as it depends on the difference between the running firmware and the firmware we are upgrading to.

2.6.2 mlxfwreset Synopsis

```
# mlxfwreset -d <device>  [--level <0..5>] [-y] <q[uery] | r[eset]>
```

where:

`-d|--device <device>`	Device to work with.
`-l|--level <0..5>`	Run reset with the specified reset level.
`-y|--yes`	Answer “yes” on prompt.
`--skip_driver, -s`	Skip driver start/stop stage (driver must be stopped manually).
`-v|--version`	Print tool version.
`-h|--help`	Show help message and exit.
`--reset_fsm_register`	Reset the fsm sync register to idle state
`--skip_fsm_sync`	Skip fsm syncing
`q|query`	Query for reset level required to load new firmware
`r|reset`	Execute reset Level.

1. In Service Firmware Update (ISFU) allows a smooth firmware upgrade.

2.6.3 Reset Levels

Reset levels depends on the extent of the changes introduced when updating the device's firmware. The tool will display the supported reset levels that will ensure the loading of the new firmware

Those reset levels are:

- 0: Full ISFU.
- 1: Driver restart (link/management will remain up).
- 2: Driver restart (link/management will be down).
- 3: Driver restart and PCI reset.
- 4: Warm reboot.
- 5: Cold reboot (performed by the user).

Full ISFU means that the transaction to the new firmware will be seamless.

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.

Driver restart includes restart of the mst driver and OFED driver.

In PowerKVM and in Linux with PPC64 and PPC64LE architectures, the tool will execute hot PCI reset.

2.6.4 mlxfwreset for Multi-Host NICs

mlxfwreset supports a Multi-Host setup. To reset the firmware for a device in a Multi-Host setup, you have to run the tool on all the hosts simultaneously. The tool utilizes a synchronization mechanism supported by the firmware in order to synchronize between the different running instances of the tool on the hosts. If the synchronization fails, when re-running the tool a message stating that the fsm register need to be reset by running the `--reset_fsm_register` command.

For debugging purposes, it is possible to avoid the synchronization by running the tool with the flag `--skip_fsm_sync`.

When running mlxfwreset on a Multi-Host setup, a time-out of 3 minutes is expected for all the hosts until they join the firmware reset process. In case a time-out has reached, you can re-run the tool using the `--reset_fsm_register` flag.

2.6.5 Examples of mlxfwreset Usage

1. To query device reset level after firmware update use the following command line:

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

Supported reset levels for loading firmware on device, /dev/mst/mt4113_pciconf0:

Example:

```
0: Full ISFU                                        Not supported
1: Driver restart (link/management will remain up)  Not supported
2: Driver restart (link/management will be down)    Not supported
3: Driver restart and PCI reset                     Supported
4: Warm Reboot                                      Supported
5: Cold Reboot                                      Supported
```

2. To reset device in order to load new firmware, use the following command line:

```
# 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.
```

When running the reset command without specifying a reset level the minimal reset level will be performed.

3. To reset a device with a specific reset level to load new firmware, use the following command line:

```
# 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       -

The system is going down for reboot NOW!
```

2.6.6 mlxfwreset Limitations

The following are the limitations of mlxfwreset:

- Executing a reset level that is lower than the minimal level (as shown in query command) will yield an error and a reboot will be required in order to load the new firmware. Attempting to run the tool again will return an error.
- When updating the firmware of a device using flint/mlxburn/mlxfwmanager, it is required for the burning tool to update the new firmware in an ISFU manner in order to be able to run this tool. The user should make sure that the message: "`-I- To load new FW run mlxfwreset or reboot machine`" is present in the burning tool's output.
- 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.

2.7 mlxphyburn – Burning Tool for Externally Managed PHY

Mlxphyburn (Mellanox PHY burn) 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.

2.7.1 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

2.7.2 mlxphyburn Synopsis

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

where:

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

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

2.7.3 Examples of mlxphyburn Usage

2.7.3.1 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.
```

2.7.3.2 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
```

2.8 mlx_fpga - Burning and Debugging Tool for Mellanox Devices with FPGA

mlx_fpga tool allows the user to burn and update a new FPGA image on Mellanox Innova adapter cards. For instructions on how to burn, please refer to section Section 2.8.3.2, “Burning the FPGA’s Flash Device using the mlx_fpga Burning Tool”, on page 115.

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

2.8.1 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 Mellanox Support at: support@mellanox
- Start mst service with the fpga lookup flag (mst start --with_fpga)

2.8.2 mlx_fpga Synopsis

```
# mlx_fpga [-d <device> ] < read <addr> | write <addr> <value> | burn <image file/s> | clear_semaphore |
reset | load | query | set_fw_mgmt
```

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

2.8.3 Examples of mlx_fpga Usage

2.8.3.1 Adding FPGA mst Device Interface

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

```
apps-13:~ # modprobe mlx5_fpga_tools
apps-13:~ # mst start --with_fpga
apps-13:~ # 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[1]
```

1. It is recommended to use the RDMA device as it faster. I2C is used for recovery purposes when RDMA is not functional.

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

```
apps-13:~ # modprobe mlx_accel_tools
apps-13:~ # mst start --with_fpga
apps-13:~ # 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[1]
```

1. It is recommended to use the RDMA device as it faster. I2C is used for recovery purposes when RDMA is not functional.

- For recovery only, where modules are broken or missing.

```
[root@apps-13 ~]# 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
[root@apps-13 ~]# 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.

2.8.3.2 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 IPsec .bin file can be downloaded from the Mellanox site (www.mellanox.con --> Products --> Adapters --> Programmable Adapter Cards --> Innova

IPsec).
The Innova Flex .bin file must be generated according to the instructions listed in the Mellanox Innova Flex adapter card User Manual.

It is recommended to burn the FPGA device using an RDMA device as it faster and it shortens the burning time.

Step 1. Burn the image.

```
# mlx_fpga -d <device> burn image.bin
```

Step 2. Load the FPGA image from flash according to Section 2.8.3.3, "Loading the Tool", on page 116 or power cycle the machine for change to take effect.

2.8.3.3 Loading the Tool

Load an FPGA image from user configurable flash:

```
# mlx_fpga -d <device> l/load <optional: load options>
```

where `<optional: load options>` is:

`--factory`	Load FPGA image from factory flash
`--user`	Load FPGA image from user flash [Default option]

2.8.3.4 Debugging the Tool

- Reading One Debug Register:

```
# mlx_fpga -d <device> read 0x0
```

- Writing One Debug Register:

```
# mlx_fpga -d <device> write 0x0 0x0
```

2.8.3.5 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:

Table 8: mlx_fpga Tool 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

Table 8: mlx_fpga Tool Registers

Name	Address	Range	Default	RW	Description
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.

2.9 cpldupdate - Tool for Programming On-Board CPLDs on Mellanox Devices

cpldupdate tool allows the user to program on-board CPLDs on supporting mellanox products. The on-board CPLDs, as well as the core engine for programing them, are provided by Lattice Semiconductors.

cpldupdate accepts VME files as input and programs the appropriate CPLD on the device. The CPLD ID is embedded in the VME file. The user need not be aware of the board composition.

A VME file is the data file for use with the ispVM Embedded programming software. The file is essentially a binary version of an SVF file. SVF commands and data are stored in a binary format (with optional compression) for efficient storage and processing by embedded microprocessors. The ispVM Embedded software, provided as source code in C, interprets the VME data to manipulate the JTAG signals of connected target devices.

cpldupdate tool was provided by Lattice Semiconductors and was modified to fit Mellanox needs.

2.9.1 Tool Requirements

- CPLD bearing board

2.9.2 cpldupdate Synopsis

```
# cpldupdate [option] vme_file [vme_file]
```

where:

--dev <device>	Specifies target lid and through which HCA device and port to send the commands. <device> (e.g. lid-N, /dev/mst/mt4115_pciconf0)
--idcode <num_of_bits>	Runs IDCODE command and exits.

2.9.3 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! |
+=======+
```

3 Debug Utilities

3.1 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.

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

3.1.1 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

Device Specific Info:

- **Connect-IB®, ConnectX®-4, ConnectX®-4 Lx, Switch-IB, Switch-IB 2, Spectrum:**

  iRISC names: [i0, iron, i2, i3, i4, i5, i6, i7, all]

- **ConnectX®-5, ConnectX®-5 Ex, BlueField:**

  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, PHY_EYE_OPN, PHY_COMMON, PHY_MANAGER, PWR, FLR, ICM_ACCESS, MAD, RXT_CHECKS, I2C, TRANSPORT, FW_LL, RX_ERRORS, CMD_DRIVER, PRO-FILING, 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
.
.
.
```

3.2 itrace Utility

The itrace utility extracts and prints trace messages generated by the firmware of a ConnectX-2/ ConnectX-2/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 Mem-Free 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.

3.2.1 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[1] or mst restart[1]).
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_MASK`	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

1. This step in not required in Windows.

`--nomap`	Sets itrace not to directly access memory (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 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 sx1
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 ddaaadc1) 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)..
```

3.3 mstdump Utility

The mstdump utility dumps device internal configuration registers. The dump file is used by Mellanox Support for hardware troubleshooting purposes. It can be applied on all Mellanox devices.

3.3.1 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.

3.4 mlxi2c Utility

The mlxi2c utility provides a way to route the I2c bus to Mellanox 4th generation (Group I) switches.

3.4.1 mlxi2c Usage

The mst driver must be started prior to running mlxi2c.

➢ ***To start mlxi2c:***

1. Start the mst driver (mst start[1] or mst restart).
2. Run mlxi2c with the following command line syntax:

```
# mlxi2c [switches...] <command> [parameters...]
```

Switches Options

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

Commands

`p <i2c_component>`	Route the i2c path to the indicated i2c component
`scan`	Scan the i2c slave addresses

Example:

- Display the addresses of all I2C-accessible devices:

```
# mlxi2c -d /dev/mst/mtusb-1 scan
```

3.5 i2c Utility

The i2c utility provides low level access to the I2C bus on any Mellanox switch platform, enabling the user to read or write data.

3.5.1 i2c Usage (Advance Users)

The mst driver must be started prior to running i2c tool. To start i2c:

1. Start the MST driver (mst start[1] or mst restart[1]).
2. Run i2c with the following command line syntax:

```
# i2c [OPTIONS] <device> <cmd> <i2c_addr> <addr> [<data>]
```

where:

`-h`	Prints this message.
`-a <addr_width>`	Sets address width (in bytes) to the specified value. May be 0, 1, 2 or 4. Default: 1.
`-d <data_width>`	Sets data width (in bytes) to the specified value. May be 1, 2 or 4s. Default is 1.
`-x <data_len>`	Presents each byte of data as two hexadecimal digits (such as 013C20343B). Note that this option is mutually exclusive with the "-d" option.

The remaining parameters are:

`<device>`	Valid mst device.
`<cmd>`	Command. May be "r[ead]" or "w[rite]".
`<i2c_addr>`	I2C slave address.
`<addr>`	Address (of length addr_width) inside I2C target device to read/write operation. Note that the <addr> value is ignored if <addr_witdh> = 0.
`<data>`	Data (bytes of length data_width) to write to target device.

All parameters are interpreted as hexadecimal values.

Examples:

1. 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
```

2. 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
```

1. This step in not required in Windows.

```
3412
```

3. 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
```

3.5.2 Exit Return Values

The following exit values are returned:

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

3.6 mget_temp Utility

The mget_temp utility reads the hardware temperature from Mellanox Technologies devices with temperature sensors (all Mellanox devices), and prints the result in Celsius degrees.

3.6.1 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.

3.7 mlxtrace Utility

The mlxtrace utility is used to configure and extract HW events generated by different units in Mellanox 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 Mellanox 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 Mellanox 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.

3.7.1 mlxtrace Usage

- The mst driver must be started prior to running the mlxtrace tool.
- For MEM buffer mode driver must be "loaded" also.
- Enter the following command:
  - mlxtrace [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 - KERNELB, UFFER (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)

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:

1. 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
```

2. 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
```

3.8 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 Mellanox 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.

3.8.1 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`	Mellanox 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*
	* Reading flow steering information is supported in ConnectX-4 and ConnectX-5 families.
`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
```

3.9 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.

3.9.1 mlxmcg Usage

mlxmcg is supported in ConnectX-3/ConnectX-3 Pro only.

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
================================================================================================
```

3.10 pckt_drop Utility

The pckt_drop utility corrupts the next transmitted packet from a ConnectX® family adapter port.

3.10.1 pckt_drop Usage

Run the pckt_drop with the following command line syntax:

`-d, --device`	Specify the mst device to configure. (Required.)
`-h, --help`	Print this help screen and exit.

`-m, --mode`	Specify operating mode. Supported modes are: **EDP**: Inserts error on next transmitted data packet. (Default: `EDP'.)
`-p, --port`	Select which port to configure. Use `1'/`2' for port1/port2, respectively, or `b' for both. (Default: `b'.)
`-v, --version`	Print the application version and exit.

Example:

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

The example above shows how to use the pckt_drop to corrupt a packet from port 1.

3.11 mlxuptime Utility

The mlxuptime is a Mellanox firmware which prints Mellanox devices' up time and measured/ configured frequency.

3.11.1 mlxuptime Usage

```
mlxuptime [options]
```

where:

`-d <dev>`	Mst device name
`-s <time>`	Sampling interval for measuring frequency (default: 1 [sec])
`-h`	Print help and exit
`-v`	Print tool version and exit

Example:

- Print all info:

```
 # mlxuptime -d /dev/mst/mt4117_pciconf0
Measured core frequency        : 427.099818 MHz
Device up time                 : 10:01:20.456344 [h:m:s.usec]
```

- Print the uptime and configured frequency only:

```
# mlxuptime -d /dev/mst/mt4117_pciconf0 -m
Configured core frequency      : 427.083333 MHz
Device up time                 : 53:31:00.162464 [h:m:s.usec]
```

3.12 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® and Connect-IB® Mellanox adapter devices.

wqdump on ConnectX®-2, ConnectX®-3 and ConnectX®-3 Pro is not supported against in-band devices.

3.12.1 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

```
 # WQDump <-d|--device DeviceName> <--source ContextType> [--gvmi Gvmi] [--qp ContextNumber]
<--dump DumpType> [--fi StartIndex] [--num NumberOfItems] [--format Format]
[--address Address] [--size Size] [-v|--version] [-h|--help] [--clear_semaphore] [--gw_access]
```

where:

`--d|--device DeviceName`	Device name
`--source ContextType`	Type of context to dump. Options are: Snd, Rcv, Cmp, Srq, Eqe, connect-X: 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 Family)

4th Generation Devices 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 Devices 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
-------------------------
```

```
 == Requestor Part (Connected) ==
Send Qpc  gvmi 0000  QP Index 0000000b
   0.   80000000   0000001e   05000000   00000000
   1.   70000000   00000000   00000000   00000000
   2.   0000000b   00ffffff   00000000   00000000
   3.   00000000   00000000   00000000   80010000
-------------------------
 == Responder Part (Not Connected (mac)) ==
Recv Qpc  gvmi 0000  QP Index 0000000b
   0.   b8eccd02   00000000   d8f5cd02   00000000
   1.   d0010000   00000000   40000000   00000000
   2.   e0e03903   00000000   f0e03903   00000000
   3.   00000000   00000000   00000000   00000000
-------------------------
```

- SRQ
  - Opened QPs:

```
# wqdump -d /dev/mst/mt4115_pciconf0 --source Srq --dump all_valid_qpns
Numbers of valid contexts (in the range 0x0 - 0xffffff):
gvmi 0x0000  index 0x00000060
gvmi 0x0000  index 0x00000061
gvmi 0x0000  index 0x00000066
```

  - Dump WQs:

```
# wqdump -d /dev/mst/mt4117_pciconf0 --source Srq --dump wq --qp 0x60
[Element Index 0]
 -------------- SRQ Next ------------------
next_wqe_index                          : 0x1
signature                               : 0x0
 -------------- scatter entry (0) ---------------
byte_count                              : 0x0
wqe_inline                              : 0x0
local_key                               : 0x0
local_address_63_32                     : 0x0
local_address_31_0                      : 0x0

[Element Index 0x1]
 -------------- SRQ Next ------------------
next_wqe_index                          : 0x0
signature                               : 0x0
 -------------- scatter entry (0) ---------------
byte_count                              : 0x0
wqe_inline                              : 0x0
local_key                               : 0x0
local_address_63_32                     : 0x0
local_address_31_0                      : 0x0
```

- CMAS
  - Opened contexts from some CMAS type (CMAS_EQ_BUFF for ex):

```
#wqdump -d /dev/mst/mt4119_pciconf0 --source CMAS_EQ_BUFF --dump ALL_VALID_QPNS
Numbers of valid contexts (in the range 0x0 - 0xffffff):
gvmi 0x0000  index 0x00000002
gvmi 0x0000  index 0x00000010
gvmi 0x0000  index 0x00000011
gvmi 0x0000  index 0x00000012
gvmi 0x0000  index 0x00000013
gvmi 0x0000  index 0x00000014
gvmi 0x0000  index 0x00000015
```

  - Dump raw data:

```
# wqdump -d /dev/mst/mt4119_pciconf0 --source CMAS_EQ_BUFF --dump QP --qp 0x15 --format
raw
CMAS  gvmi 0000  CMAS Index 00000015
0.   80000002   00000000   00000000   e4f80000
1.   00000000   00000000   00000000   00000000
2.   00000000   00000000   00000000   00000000
3.   00000000   00000000   00000000   00000000
--------------------------
```

3.13 mlxmdio Utility

The mlxmdio tool is used to read/write MDIO registers (Clause 45) on Boards with externally managed PHY.

3.13.1 mlxmdio Usage

To run mlxmdio, use the following line:

```
# mlxmdio <-d mst_dev> <-m phy_addr:dev_addr> <-a addr[:data]> [-g mdio_gw]
```

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 gw <0 or 1> to use. (Default is 0).
`-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) if the firmware interface is not supported by the device, the tool by default will attempt to send the transaction through the mdio gateway (default is 0). If -g <0|1>

flag is specified, the tool will attempt to send the MDIO transaction through the specified gateway.

Sending MDIO transactions via FW requires specification of the PCI device.

Example:

1. To read mlxmdio register, run the following command:

```
# mlxmdio -d /dev/mst/mt4099_pciconf0 -m 0x2:0x1 -a 0x0
4081
```

2. To write mlxmdio register, run the following command:

```
# mlxmdio -d /dev/mst/mt4099_pciconf0 -m 0x2:0x1e -a 0x103:0x1
```

3. To read mlxmdio register through gateway, run the following command:

```
# mlxmdio -d /dev/mst/mt4099_pciconf0 -m 0x0:0x1 -a 0x3 -g 0
688b
```

3.14 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 Mellanox support before using it.

3.14.1 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:

```
mlxreg -d /dev/mst/mt4115_pciconf0 --show_regs

Available Access Registers
================================================================================
MCIA
MPCNT
MPEIN
PAOS
PDDR
PLIB
PMLP
PPAOS
PPCNT
PPLM
PPLR
PPRT
PPTT
PTYS
SLRG
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...
```

3.15 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).

mlxlink is intended for advanced users with appropriate technical background.

When using mlxlink to disable the port state ("`--port_state dn`" flag) on a NIC connected through a Socket-Direct connection, the port must be disabled on both mst devices representing the physical port.

3.15.1 mlxlink Usage

To run mlxlink:

```
mlxlink [OPTIONS]
```

where:

`--ber_limit <limit_criteria>`	BER Limit Criteria [Nominal(Default)/Corner/Drift] (Optional - Default Nominal)
`--database`	Save Transmitter Configuration for Current Speed Permanently (Optional)
`--link_mode_force`	Configure Link Mode Force (Disable AN)
`--fec_speed <fec_speed>`	Speed to Configure FEC [100G/50G/25G/...] (Default is Active Speed)
`--twisted_pair_force_mode <twisted_pair_force_mode>`	Twisted Pair Force Mode [MA(Master)/SL(Slave)]
`--iteration <iteration>`	Iteration Number of BER Collection
`--pc`	Clear Counters

`--port_type <port_type>`	Port Type [NETWORK(Default)/PCIE/OOB]
`--rx_prbs <rx_prbs_mode>`	RX PRBS Mode [PRBS31(Default)/PRBS7/...] (Optional - Default PRBS31)
`--rx_rate <rx_lane_rate>`	RX Lane Rate [EDR(Default)/25G/10G/...] (Optional - Default 25G)
`--serdes_tx <params>`	Configure Transmitter Parameters [polarity,ob_tap0,...]
`--serdes_tx_lane <transmitter_lane>`	Transmitter Lane to Set (Optional - Default All Lanes)
`--show_ber_monitor`	Show BER Monitor Info
`--show_device`	General Device Info
`--show_fec`	Show FEC Capabilities
`--show_serdes_rx`	Show Receiver Info
`--show_serdes_tx`	Show Transmitter Info
`--test_mode <prbs_mode>`	Physical Test Mode Configuration [EN(enable)/DS(disable)/ TU(perform tuning)]
`--tx_prbs <tx_prbs_mode>`	TX PRBS Mode [PRBS31(Default)/PRBS7/...] (Optional - Default PRBS31)
`--tx_rate <tx_lane_rate>`	TX Lane Rate [EDR(Default)/25G/10G/...] (Optional - Default 25G)
`-a |--port_state <port_state>`	Configure Port State [UP(up)/DN(down)/TG(toggle)]
`-b |--ber_collect <csv_file>`	Port Extended Information Collection [CSV File]
`-c |--show_counters`	Show Physical Counters and BER Info
`-d |--device <device>`	Perform operation for a specified mst device
`-e |--show_eye`	Show Eye Opening Info
`-h |--help`	Display help message.
`-k |--fec <fec_override>`	Configure FEC [AU(Auto)/NF(No-FEC)/FC(FireCode FEC)/ RS(RS-FEC)]
`-l |--loopback <loopback>`	Configure Loopback Mode [NO(No Loopback)/PH(phy loopback)/EX(external loopback)]
`-m |--show_module`	Show Module Info
`-p |--port <port_number>`	Port Number
`-s |--speeds <speeds>`	Configure Speeds [speed1,speed2,...]
`-v |--version`	Display version info.

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
```

- Set Auto Negotiation to OFF:

```
mlxlink -d <device> -p <port_number> --speeds <speed> --link_mode_force
```

- 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)
```

- Perform PRBS Tuning:

```
mlxlink -d <device> -p <port_number> --test_mode TU
```

- Configure Transmitter Parameters (on lane, to database):

```
mlxlink -d <device> -p <port_number> --serdes_tx <polar-
ity>,<ob_tap0>,<ob_tap1>,<ob_tap2>,<ob_bias>,<ob_preemp_mode>,<ob_reg>,<ob_leva> (--serdes_tx-
_lane <lane number>) (--database)
```

3.15.2 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 Mellanox NIC, to address port 2, use `mlxlink -d /dev/mst/mt4115_pciconf0 --gvmi_address <0xAddress>`.

4 Cables Utilities

MFT can work against the cables that are connected to the devices on the machine, or in the InfiniBand fabric in the following scenarios:

- Accessing the cable using the local PCI device.
  Supported in:
  - ConnectX-4 and newer devices
  - All the platforms
- Accessing the cable using the IB fabric.
  Supported in:
  - All the devices
  - Linux and Windows since IB driver is required
- Supported cables are all QSFP and SFP

4.1 Cables Discovery

4.1.1 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.

Example:

➢ ***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
```

4.1.2 Representing the Cables in mst Status

4.1.2.1 Local PCI 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
```

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
```

4.2 Working with Cables

The following are the tools that can work with cables: mstdump and mlxdump.

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
```

4.3 mlxcables - Mellanox Cables Tool

The mlxcables tool allows users to access the cables and do the following:

- Query the cable and get its IDs
- Read specific addresses in the EEPROM
- Read a specific register by its name. Supported registers are received by the tool (depends on the cable type)
- Dump all the cable EEPROM bytes in RAW format
- Upgrade the FW image on the cable uC (Only on cables that support ISFU)

4.3.1 mlxcables Synopsis

```
[-d|--dev <DeviceName>] [-h|--help] [-v|--version] [-q|--query] [--DDM] [-r|--read]
[--print_raw] [--dump] [-b|--bytes_line <bytesPerLine>] [-p|--page <pageNum>] [-o|--offset
<pageOffset>] [-l|--length <length>] [-a|--address <address>] [-u|--update] [-i|--image-file
<FileName>] [-f|--force] [-y|--yes] [--no] [--read_reg <Register>] [--read_all_regs]
[--show_all_regs] [--customization <Customization>]
```

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)
`-u|--update`	Update firmware image on the cable
`-i|--image-file <FileName>`	Specified image file to use

`-f|--force`	Force image update
`-y|--yes`	Answer is yes in prompts
`--no`	Answer is no in prompts
`--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>`	Show customization parameters
`--customization <customization_type>`	Show cable specific customization

Notes:

- For QSFP transceivers, the tool reads the address from I2C address of 0x50. For further information, please see spec SFF8636.
- For SFP transceivers, the tool reads from I2C address 0x50 and names it page 0. When reading from I2C address 0x51 the pages will be read as page <x+1>, for example:
  - I2C address 0x51 page 0 will be referred in the tool as page 1.
  - I2C address 0x51 page 1 will be referred in the tool as page 2.
    For further information, please see spec SFF8472.

Examples:

➢ ***To read specific byte/s in the cable pages:***

```
mlxcables -d mt4115_pciconf0_cable_0 -r -p 0 -o 165 -l 3
Page[0].Byte[165] = 0x00
Page[0].Byte[166] = 0x02
Page[0].Byte[167] = 0xc9
```

Another way to read from a specific page is to use the `'--address <ADDR>'` flag where `ADDR=0x<PAGE><OFFSET>`, for example to read the same bytes with `-a`:

```
mlxcables -d mt4115_pciconf0_cable_0 -r -a 0x00A5 -l 3
Page[0].Byte[165] = 0x00
Page[0].Byte[166] = 0x02
Page[0].Byte[167] = 0xc9
```

➢ ***To read in raw format:***

```
# mlxcables -d mt4115_pciconf0_cable_0 -r -p 0 -o 128 -l 12 --print_raw

128: 0d 8c 23 81
132: 00 00 00 00
136: 00 00 00 05
to control Bytes per line, use -b:
# mlxcables -d mt4115_pciconf0_cable_0 -r -p 0 -o 128 -l 12 --print_raw -b 8

128: 0d 8c 23 81 00 00 00 00
136: 00 00 00 05
```

➢ ***To query the cable:***

```
mlxcables -d mt4115_pciconf0_cable_0 -q
Cable name    : mt4115_pciconf0_cable_0
FW version    : 2.2.550
FW Dev ID     : 0x21
FW GW version : Legacy
-------- Cable EEPROM --------
Identifier    : QSFP+ (0dh)
Technology    : 1550 nm DFB (50h)
Compliance    : 40G Active Cable (XLPPI), 100G AOC or 25GAUI C2M AOC. Providing a worst
BER of 10^(-12) or below
Wavelength    : 1550 nm
OUI           : 0x0002c9
Vendor        : Mellanox
Serial number : MT1602FT00022
Part number   : MFS1200-E003
Revision      : AB
Temperature   : 54 C
Length        : 3 m
```

➢ ***Get the DDM query of the cable:***

```
# mlxcables -d mt4115_pciconf0_cable_0 --DDM
Cable DDM:
----------
Temperature    : 54C
Voltage        : 3.2527V
Channel 1:
        RX Power : -3.3442dBm
        TX Power : -infdBm
        TX Bias  : 113.5420mA
Channel 2:
        RX Power : -1.1182dBm
        TX Power : -infdBm
        TX Bias  : 113.7360mA
Channel 3:
        RX Power : -3.1515dBm
        TX Power : -infdBm
        TX Bias  : 102.2800mA
Channel 4:
        RX Power : -4.0894dBm
        TX Power : -infdBm
        TX Bias  : 102.2800mA
----- Thresholds -----
Temperature:
        High Warning  : 70C
```

```
        Low  Warning  : 0C
        High Alarm    : 75C
        Low  Alarm    : -5C
        Waring mask   : 0
        Alarm mask    : 0
Voltage:
        High Warning : 3.4650V
        Low  Warning : 3.1350V
        High Alarm   : 3.6300V
        Low  Alarm   : 2.9700V
        Waring mask  : 0
        Alarm mask   : 0
Channel 1:
        RX Power high warn   : 3.0103dBm
        RX Power low  warn   : -10.0000dBm
        RX Power high alarm  : 3.9742dBm
        RX Power low  alarm  : -10.9691dBm
        RX Power Waring mask : 0
        RX Power Alarm mask  : 0
        TX Power high warn   : 1.7609dBm
        TX Power low  warn   : -6.9897dBm
        TX Power high alarm  : 3.0103dBm
        TX Power low  alarm  : -10.0000dBm
        TX Power Waring mask : 0
        TX Power Alarm mask  : 0
        TX Bias high warn    : 0.0000mA
        TX Bias low  warn    : 0.0000mA
        TX Bias high alarm   : 0.0000mA
        TX Bias low  alarm   : 0.0000mA
        TX Bias Waring mask  : 0
        TX Bias Alarm mask   : 0
Channel 2:
        RX Power high warn   : 3.0103dBm
        RX Power low  warn   : -10.0000dBm
        RX Power high alarm  : 3.9742dBm
        RX Power low  alarm  : -10.9691dBm
        RX Power Waring mask : 0
        RX Power Alarm mask  : 0
        TX Power high warn   : 1.7609dBm
        TX Power low  warn   : -6.9897dBm
        TX Power high alarm  : 3.0103dBm
        TX Power low  alarm  : -10.0000dBm
        TX Power Waring mask : 0
        TX Power Alarm mask  : 0
```

```
        TX Bias high warn    : 0.0000mA
        TX Bias low  warn    : 0.0000mA
        TX Bias high alarm   : 0.0000mA
        TX Bias low  alarm   : 0.0000mA
        TX Bias Waring mask  : 0
        TX Bias Alarm mask   : 0
Channel 3:
        RX Power high warn   : 3.0103dBm
        RX Power low  warn   : -10.0000dBm
        RX Power high alarm  : 3.9742dBm
        RX Power low  alarm  : -10.9691dBm
        RX Power Waring mask : 0
        RX Power Alarm mask  : 0
        TX Power high warn   : 1.7609dBm
        TX Power low  warn   : -6.9897dBm
        TX Power high alarm  : 3.0103dBm
        TX Power low  alarm  : -10.0000dBm
        TX Power Waring mask : 0
        TX Power Alarm mask  : 0
        TX Bias high warn    : 0.0000mA
        TX Bias low  warn    : 0.0000mA
        TX Bias high alarm   : 0.0000mA
        TX Bias low  alarm   : 0.0000mA
        TX Bias Waring mask  : 0
        TX Bias Alarm mask   : 0
Channel 4:
        RX Power high warn   : 3.0103dBm
        RX Power low  warn   : -10.0000dBm
        RX Power high alarm  : 3.9742dBm
        RX Power low  alarm  : -10.9691dBm
        RX Power Waring mask : 0
        RX Power Alarm mask  : 0
        TX Power high warn   : 1.7609dBm
        TX Power low  warn   : -6.9897dBm
        TX Power high alarm  : 3.0103dBm
        TX Power low  alarm  : -10.0000dBm
        TX Power Waring mask : 0
        TX Power Alarm mask  : 0
        TX Bias high warn    : 0.0000mA
        TX Bias low  warn    : 0.0000mA
        TX Bias high alarm   : 0.0000mA
        TX Bias low  alarm   : 0.0000mA
        TX Bias Waring mask  : 0
        TX Bias Alarm mask   : 0
```

➢ ***To read by register name:***

1. Get the list of the supported registers.

```
mlxcables -d mt4115_pciconf0_cable_0 --show_all_regs
Available registers per page:
================================================
page00_high registers:
================================================
identifier                |
ext_identifier            |
connector_type             |
..
..
..
vendor_oui                 |
..
```

2. Read the register with the register name you choose (e.g. vendor_oui, identifier).

```
mlxcables -d mt4115_pciconf0_cable_0 --read_reg vendor_oui
vendor_oui = 0x0002c9
mlxcables -d mt4115_pciconf0_cable_0 --read_reg identifier
identifier = 0x0d
```

➢ ***To read all the Eeprom of the cable:***

```
mlxcables -d mt4115_pciconf0_cable_0 --read_all_regs
Available registers per page:
================================================
page00_high registers:
================================================
identifier               |       0x0d
ext_identifier           |       0x06
connector_type           |       0x02
..
..
..
vendor_oui               |       0x0002c9
..
```

This will print the same tables as "`--show_all_regs`" but with the data that was read.

➢ ***To dump all the cable's pages in raw format:***

```
# mlxcables -d mt4115_pciconf0_cable_0 --dump -b 16

+---------------------------------------+
| Page: 0x00 , Offset: 000, Length: 0x80 |
+---------------------------------------+
```

```
000: 0d 06 00 f0 00 00 00 00 00 00 00 00 00 00 00 00
016: 00 00 00 00 00 00 29 a7 00 00 80 7d 00 00 00 00
032: 00 00 37 fa 26 2a 33 68 1c 0c b3 f9 b2 76 c2 fc
048: c1 3e 00 00 00 00 00 00 00 00 00 00 00 00 00 00
064: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
080: 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00
096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 00
112: 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00

+----------------------------------------+
| Page: 0x00 , Offset: 128, Length: 0x80 |
+----------------------------------------+

128: 0d cc 23 81 00 00 00 00 00 00 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
```

4.3.2 MTUSB Cable Board

The mlxcables tool supports reading cable data directly via i2c when the cable is connected to a dedicated board. The board is connected to the host with an MTUSB adapter.

Examples on a Windows machine:

- After adding the cables using `'mst cable add'` the following mst status is presented:

```
mst status
MST devices:
------------
```

```
  mtusb-1

Cable MST devices:
------------------
  mtusb-1_cable
```

- Query the cable:

```
mlxcables -d mtusb-1_cable
Querying Cables ....

Cable #1:
---------
Cable name    : mtusb-1_cable
FW version    : 2.0.208
FW Dev ID     : 0x20
FW GW version : Legacy
-------- Cable EEPROM --------
Identifier    : QSFP+ (0dh)
Technology    : 850 nm VCSEL (00h)
Compliance    : 40G Active Cable (XLPPI), 100G AOC (Active Optical Cable) or 25GAUI C2M AOC.
Wavelength    : 850 nm
OUI           : 0x0002c9
Vendor        : Mellanox
Serial number : MT1707FT01544
Part number   : MFA1A00-E001
Revision      : A1
Temperature   : 31 C
Length        : 1 m
```

- Read from a specific address:

```
mlxcables -d mtusb-1_cable -r -p 0 -o 165 -l 3
Page[0].Byte[165] = 0x00
Page[0].Byte[166] = 0x02
Page[0].Byte[167] = 0xc9
```

4.3.3 Cable Firmware Upgrade with In Service Firmware Update (ISFU)

In Service Firmware Update (ISFU) enables a seamless update of the cable firmware.

4.3.3.1 Prerequisites

- The cable firmware gateway version must be set to ISFU (see `"FW GW version : ISFU"` in the example below).
- Burn the cable firmware image binary.

```
# mlxcables -d mt52000_pciconf0_cable_22 -i img.bin -u
```

To get the cable's firmware binary, please contact Mellanox Support (support@mellanox.com).

Burning Process Example:

- Query the device:

```
# mlxcables -d mt52000_pciconf0_cable_22
Querying Cables ....

Cable #1:
---------
Cable name    : mt52000_pciconf0_cable_22
FW version    : 32.20.121
FW Dev ID     : 0x22
FW GW version : ISFU
-------- Cable EEPROM --------
Identifier    : QSFP28 (11h)
Technology    : 850 nm VCSEL (00h)
Compliance    : Extended Specification Compliance is valid, 100GBASE-SR4 or 25GBASE-SR
Wavelength    : 850 nm
OUI           : 0x0002c9
Vendor        : Mellanox
Serial number : MT1623FT01610
Part number   : MMA1B00-C100D
Revision      : A2
Temperature   : 31 C
Length        : 50 m
```

- Query the image:

```
# mlxcables -i img.bin
Image name    : img.bin
FW version    : 32.30.204
FW Dev ID     : 0x22
FW GW version : ISFU
```

- Upgrade the firmware:

```
# mlxcables -i img.bin -d mt52000_pciconf0_cable_22 -u
Cable FW version    : 32.20.121
Image FW version    : 32.30.204
The image version is newer than the FW version on the cable, Do you want to proceed (y/N)?
y
100%
-I- The FW was updated successfully
```

- Query and verify the new firmware version:

```
# mlxcables -d mt52000_pciconf0_cable_22
Querying Cables ....

Cable #1:
---------
Cable name    : mt52000_pciconf0_cable_22
FW version    : 32.30.204
FW Dev ID     : 0x22
FW GW version : ISFU
-------- Cable EEPROM --------
Identifier    : QSFP28 (11h)
Technology    : 850 nm VCSEL (00h)
Compliance    : Extended Specification Compliance is valid, 100GBASE-SR4 or 25GBASE-SR
Wavelength    : 850 nm
OUI           : 0x0002c9
Vendor        : Mellanox
Serial number : MT1623FT01610
Part number   : MMA1B00-C100D
Revision      : A2
Temperature   : 31 C
Length        : 50 m
```

5 Troubleshooting

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

5.1 General Related Issues

Table 9: 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 Mellanox 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 Mellanox 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: • 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.

5.2 mlxconfig Related Issues

Table 10: mlxconfig Related Issues

Issue	Cause	Solution
Server not booting after enabling SRIOV with high number of VFs	Setting number of VFs larger than what the Hardware and Software can support may cause the system to cease working	To solve this issue: 1. Disable SRIOV in bios 2. Reboot server 3. Change num of VFs 4. Enable SRIOV in bios
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 mlxconfig on 5th generation (Group II) devices, the configuration's value does not change	Firmware loads the default configuration only upon reboot	Reboot the server

5.3 Installation Related Issues

Table 11: Installation Related Issues

Issue	Cause	Solution
Unable to install MFT package on ESXi platform and the following message is printed on the screen: `Got no data from process`	Insufficient privileges	1. Copy the MFT 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 MFT package **Note**: an additional reboot will be required after completing the installation
Unable to install kernel-mft in Linux due to compilation error that contains the following message: `'error: conflicting types for 'compat_sigset_t''`	CONFIG_COMPAT might not be enabled in the kernel configuration.	Set the CONFIG_COMPAT to "`y`" in the kernel .config file.

5.4 Firmware Burning Related Issues

Table 12: Firmware Burning Related Issues

Issue	Cause	Solution

The following message is printed on screen when performing firmware update:
`An update is needed for the flash layout.`
`The Operation is not failsafe and terminating the process is not allowed.`
	A flash alignment operation is required.	Approve the alignment, avoid process interrupt.

Firmware update fails with the following message: `-E- Burning FS4 image failed: Bad parameter` **Note:** This is a rare scenario.	Firmware compatibility issue.	Re-run the burn command with `--no_fw_ctrl` flag.
The following message is printed on screen when performing firmware update: `Shifting between different image partition sizes requires current image to be re-programmed on the flash. Once the operation is done, reload FW and run the command again` **Note:** This is a rare scenario.	Firmware compatibility issue.	Re-load firmware and re-run the burn command.
The following message is printed on screen when trying to query/burn a Connect-IB device: `-E- Cannot open Device: /dev/mst/mt4113_pciconf0. B14 Operation not permitted MFE_CMDIF_GO_BIT_BUSY`	Using an outdated firmware version with the Connect-IB adapter.	1. Unload MLNX_OFED driver: `/etc/init.d/openibd stop.` 2. Add “-ocr” option to the 'flint' command. For example: `flint -d /dev/mst/mt4113_pciconf0 -ocr q`
The following message is reported on screen when trying to remove the expansion ROM using the 'drom' option: `-E- Remove ROM failed: The device FW contains common FW/ROM Product Version - The ROM cannot be removed separately.B9`	Updating only the EXP_ROM (FlexBoot) for recent firmware images which requires adding the 'allow_rom_change' option.	Allow “-allow_rom_change” option to the “flint” command. For example: `flint -d <mst_device> -allow_rom_change drom`

Table 12: Firmware Burning Related Issues

Issue	Cause	Solution

Generating a firmware image file on Windows fails and the following message is printed on screen:
`-E- File: C:/Users/Administrator/ps.ini, Line: 1 - Invalid syntax`
`-E- Image generation failed: child process exited abnormally`
	Using a firmware configuration file (.ini) which was generated by PowerShell text redirection: `flint -d <mst_device> dc > <fw_conf_file>.ini`	Generate the firmware configuration file (.ini) using CMD edit and continue with generating the firmware image file.

Burning command fails and the following message is printed on screen: `-E- Can not open 06:00.0: Can not obtain Flash semaphore (63). You can run "flint -clear_semaphore -d <device>" to force semaphore unlock. See help for details.`	Semaphore can be locked for any of the following reasons: • Another process is burning the firmware at the same time • Failure in the firmware boot • Burning process was forcefully killed • In a Multihost environment, another Host is currently burning the firmware	If no other process is taking place at the same time run the following command: `flint -d <device> --clear_semaphore` OR Reboot the machine.
Burning tool fails with the following message: `-E- Unsupported binary version (2.0) please update to latest MFT package.`	The binary version is incompatible with the burning tool.	Update MFT to the latest package.
mlxburn tool fails to generate a firmware image and displays the following message: `-E- Unsupported MLX file version (2.0) please update to latest MFT package.`	The MLX file version is incompatible with the image generation tool (mlxburn).	Update MFT to the latest package.
mlxburn tool fails to generate a firmware image and displays the following message `-E- Perl Error: Image generation tool uses mic (tool) version 1.5.0 that is not supported for creating a bin file for this FW version. FW requires mic version 2.0.0 or above. Please update MFT package.`	The MLX file version is incompatible with the image generation tool (mlxburn).	Update MFT to the latest package.

Burning tool fails with an error mentioning Firmware time stamping
e.g
`-E- Burning FS3 image failed: Stamped FW version missmatch: 12.16.0212 differs from 12.16.0230`
	The device was set with a timestamp for a different firmware version than the one being burnt or the image is stamped with an older timestamp	Either set a newer timestamp on the image than there is on the device, or reset the timestamp completely.
		`flint -d <device> ts reset`
		`flint -i <image> ts reset`

Table 12: Firmware Burning Related Issues

Issue	Cause	Solution
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.

5.5 Secure Firmware Related Issues

Table 13: 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 Mellanox 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 Mellanox Support to receive a signed firmware image.
Burning tool fails with the following error: `-E- Burning FS3 image failed: Component is not applicable.`	The image does not match the device (Wrong ID).	Contact Mellanox 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 Mellanox Support to receive a signed firmware image.
Burning tool fails with the following error: `-E- Burning FS3 image failed: There is no Debug Token installed.`	The debug firmware was burnt before the debug token was installed on the device.	Install the debug token using mlxconfig and then re-burn the firmware.
Burning firmware on a secure device fails with one of the following messages: • `-E- Burning FS3 image failed: Rejected authentication` • `The FW image is not secured` • `The key is not applicable`	The image was not secured in a the proper way.	Ask for a secure image with the right keys that match the device.
Secure Firmware fails when using flint brom and drom commands.	flint brom and drom commands are not 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

Table 13: Secure Firmware Related Issues

Issue	Cause	Solution
Applying token on the device fails with one of the following messages: • `Component is not applicable` • `The manufacturing base MAC was not listed` • `Mismatch FW version` • `Mismatch user timestamp` • `Rejected forbidden version`	The token was not generated or signed in the proper way.	Refer to the section E.2 "Create Tokens for Secure Firmware and NV LifeCycle," on page 183 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"`.

Appendix A: Assigning PSID

In some cases, OEMs or board manufacturers may wish to use a specific FW configuration not supplied by Mellanox. 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.

A.1 PSID Field Structure

The PSID field is a 16-ascii (byte) character string. If the assigned PSID length is less then 16 characters, the remaining characters are filled with binary 0s by the burning tool.

Table 14 provides the format of a PSID.

Table 14: PSID format

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 MHXL-CF128-T HCA board is MT_0030000001, where:

MT_	Mellanox vendor symbol
003	MHXL-CF128-T board symbol
000	Board version symbol
0001	Parameter Set Number

A.2 Assigning PSID and Integrat 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 assure PSID uniqueness.
   If you do not know your vendor symbol, please contact your local Mellanox FAE.
3. Set the PSID parameter in the new FW configuration file.

Appendix B: Remote Access to Mellanox Devices

B.1 Burning a Switch In-band Device Using mlxburn

In order to update MT52000 Switch-IB™ device with a specific GUID (for example, 0xe-41d2d03001094b0) 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.

Step 3. 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  ]
```

Step 4. Make sure the local ports are active by running 'ibv_devinfo'.

Step 5. Obtain the device LID. There are two ways to obtain it:

- 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
  ```

  To list the discovered mst inband devices run "mst status".

  ```
  [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]>
  ```

- 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.

Step 6. Run *mlxburn* with the LID retrieved in Step 5 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.
```

B.2 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.

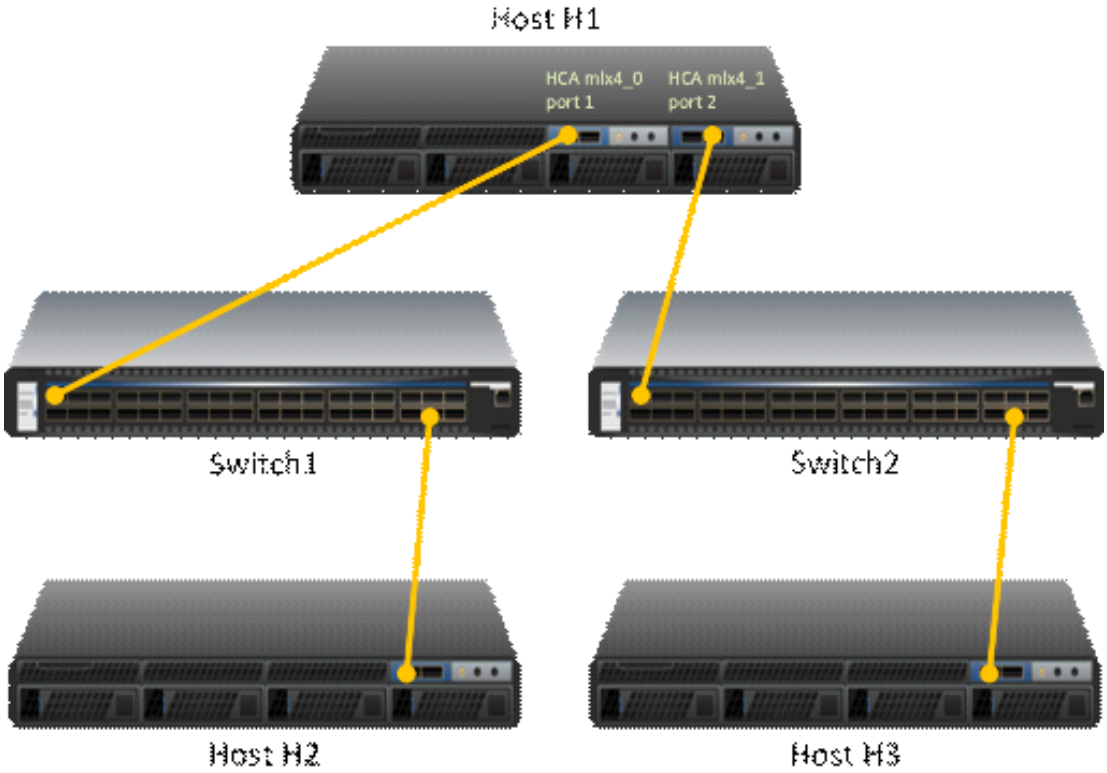
The device name would be in the format:

```
<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>
```

Example:

1. Add nodes of both subnets, Run:

```
# mst ib add mlx4_0 1
# mst ib add mlx4_1 2
```

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.

B.3 MTUSB-1 USB to I2C Adapter

The MTUSB-1 is a USB to I^2C bus adapter. This chapter provides the user with hardware and software installation instructions on machines running Linux or Windows operating systems.

Figure 4: MTUSB-1 Device

B.3.1 Package Contents

Please make sure that your package contains the items listed in Table 15 and that they are in good condition.

Table 15: MTUSB-1 Package Contents

Item	Quantity	Description
MTUSB-1 device	1	USB to I^2C bus adapter
USB cable	1	USB_A to USB_B (1.8m)
I2C cable	1	9-pin male-to-male cable (1.5m)
Converter cable	2	9-pin female to 3-pin (small/large) (0.3m)

B.3.2 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.

B.3.3 Supported Platforms

MTUSB-1 is supported in Linux and Windows only.

B.3.4 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.

B.3.5 Software Installation

The MTUSB-1 device requires that the Mellanox Firmware Tools (MFT) package be installed on the machine to which MTUSB-1 is connected – see Section 1.3,“MFT Installation,” on page 24 of this manual for installation instructions.

For a Windows machine, it is also required to install the MTUSB-1 driver – visit http://www.dio-lan.com to download this driver. This driver is required for the first use of the MTUSB-1 device.

Once you have the requirements installed, you may verify that your MTUSB-1 device is detected by MFT software as described below.

1. Start the *mst*[1] driver. Enter:

```
# mst start          (or mst restart if mst start was run earlier)
```

2. To obtain the list of *mst* devices, enter:

```
# mst status
```

1. This step in not required in Windows.

If MTUSB-1 has been correctly installed, **"mst status"** should include the following device in the device list it generates:

- On Linux: /dev/mst/mtusb-1
- On Windows: mtusb-1

B.3.6 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.

B.4 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]	• Establishes connection with a specified host on a specified port (default port is 23108). • Adds devices on remote peer to local the devices list. • <hostname> may be host name as well as an IP address.
mst remote del <host-name>[:port]	Removes all remote devices on a specified hostname. <host-name>[:port] should be specified exactly as in the "mst remote add" command.

Example:

The example below shows how to query the firmware of a device in the server side (machine: mft) from the client side (machine: mft1):

1. Run mst status in the server side:

```
[root@mft ~]# mst status
MST modules:
------------
    MST PCI module loaded
    MST PCI configuration module loaded

MST devices:
------------
/dev/mst/mt4099_pciconf0        - PCI configuration cycles access.
                                  domain:bus:dev.fn=0000:0b:00.0 addr.reg=88 data.reg=92
                                  Chip revision is: B0
/dev/mst/mt4099_pci_cr0         - PCI direct access.
                                  domain:bus:dev.fn=0000:0b:00.0 bar=0xd2600000 size=0x100000
                                  Chip revision is: B0
/dev/mst/mtusb-1:                - USB to I2C adapter as I2C master
```

2. Start the mst server in the 'server side':

```
[root@mft ~]# mst server start
```

3. Add mst remote device in the client side:

```
[root@mft1 ~]# mst remote add mft
```

4. Show the mst device in the 'client side' which contains remote devices for the 'server side' machine:

```
[root@mft1 ~]# mst status
MST modules:
------------
    MST PCI module loaded
    MST PCI configuration module loaded

MST devices:
------------
/dev/mst/mt4099_pciconf0         - PCI configuration cycles access.
                                   domain:bus:dev.fn=0000:0b:00.0 addr.reg=88 data.reg=92
                                   Chip revision is: 01
/dev/mst/mt4099_pci_cr0          - PCI direct access.
                                  domain:bus:dev.fn=0000:0b:00.0 bar=0xd2600000 size=0x100000
                                   Chip revision is: 01

Remote MST devices:
-------------------
/dev/mst/mft:23108,@dev@mst@mt4099_pciconf0
                                   Chip revision is: B0
/dev/mst/mft:23108,@dev@mst@mt4099_pci_cr0
                                   Chip revision is: B0
/dev/mst/mft:23108,@dev@mst@mtusb-1
```

5. Access a remote mst device from the 'client side':

```
[root@mft1 ~]# flint -d /dev/mst/mft:23108,@dev@mst@mt4099_pci_cr0 q
Image type:      FS2
FW Version:      2.32.1092
FW Release Date: 17.8.2014
Rom Info:        type=PXE version=3.5.305 cpu=AMD64
Device ID:       4099
Description:     Node             Port1            Port2            Sys image
GUIDs:           0002c90300e6e4e0 0002c90300e6e4e1 0002c90300e6e4e2 0002c90300e6e4e3
MACs:                                 0002c9e6e4e1     0002c9e6e4e2
VSD:             n/a
PSID:            MT_1090120019
```

B.5 Accessing Remote InfiniBand Device by Direct Route MADs

➢ ***To access IB devices remotely by direct route MADs (except for Connect-X3 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.
```

Appendix C: 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.[1]

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.

C.1 Booting Card in Livefish Mode

➢ ***To boot the card in livefish mode:***

Step 1. Power off the machine.

Step 2. Locate the Flash preset pins on the HCA.

Step 3. Close the two pins using a jumper.

Step 4. Power on the machine.

C.2 Booting Card in Normal Mode

➢ ***To boot the card in normal mode:***

Step 1. Power off the machine.

Step 2. Take off the jumper connected to the Flash Present pins on the HCA.

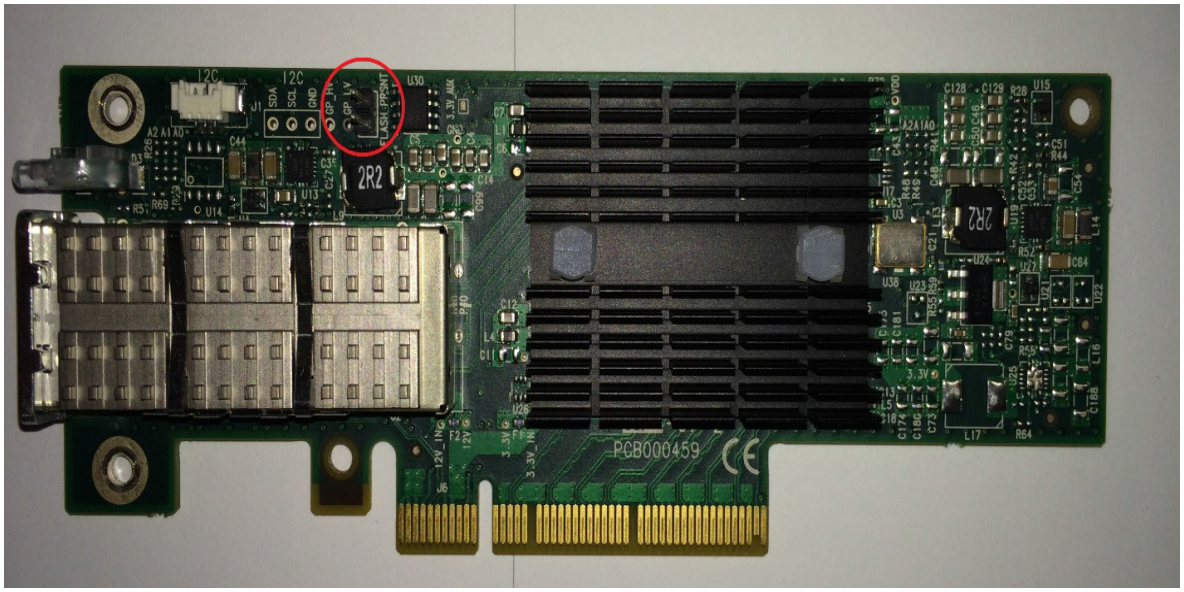
Step 3. Power on the machine.

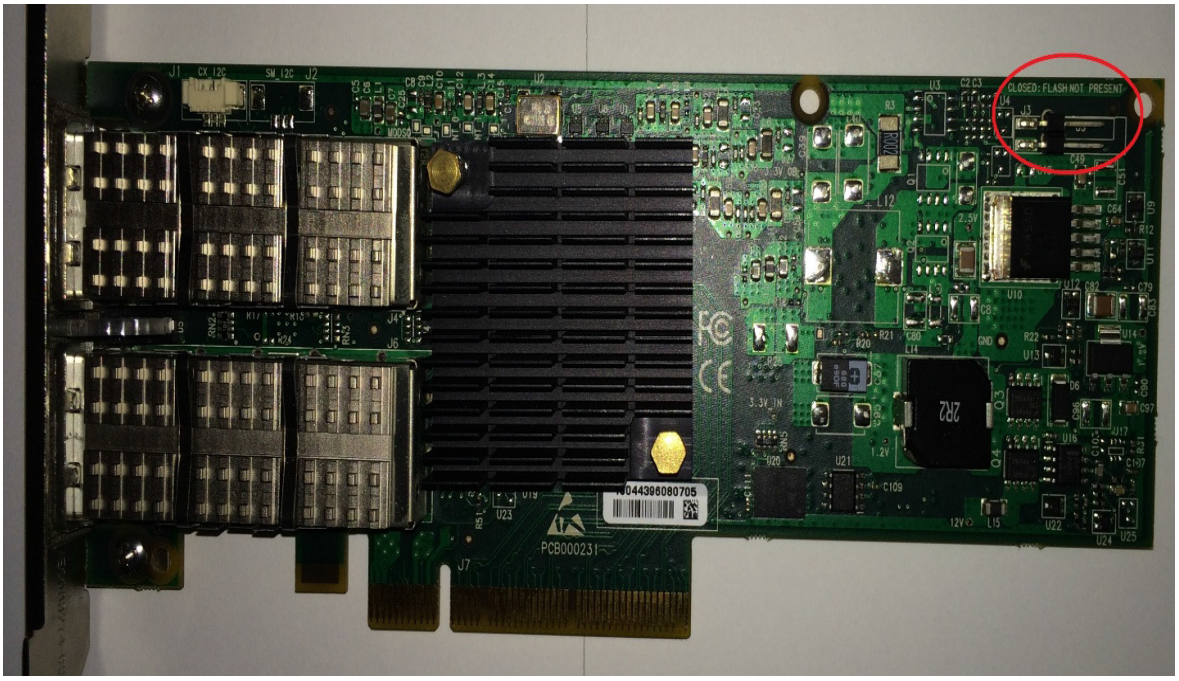
C.3 Common Locations of Flash Present Pins

The following photos show common locations of the Flash Present pins.

Existence and location of Flash Present pins depends on the board manufacture

1. Supported boards only


I2C
SDA
SCL
GND
FLASH_PRSNT
2R2
PCB000459


CLOSED: FLASH NOT PRESENT
2R2
PCB000231

Appendix D: Burning a New Device

D.1 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.

D.1.1 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 Mellanox methodology:

Description:	UID	Number	Step
Port1 GUID:	base	8	1
Port2 GUID:	base + 8	8	1
Port1 MAC:	guid2mac[1](base)	8	1
Port2 MAC:	guid2mac(base + 8)	8	1

1. guid2mac(guid) is (((guid >> 16) & 0xffffff000000) | (guid & 0xffffff)). Meaning, remove the 2 middle bytes of an 8 bytes GUID to generate a 6 bytes MAC.

D.1.2 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

D.1.3 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 the flint:

   For devices using Winbond flash:.

```
# flint -d /dev/mst/mt511_pciconf0 -ocr hw set Flash0.WriteProtected=Top,1-SubSectors
```

5. Enable flash quad SPI IO operations.

```
# flint -d /dev/mst/mt511_pciconf0 -ocr hw set QuadEn=1
```

Method 2: Generating Firmware Image With Blank GUIDs, Burning and Setting GUIDs on the Flash

1. Generate the initial image with VPD for the specific device, using the mlxburn tool. The generated image occupies full flash size.

```
# mlxburn -fw FW/fw-ConnectIB.mlx -c FW/MCB194A-FCA_A1.ini -wrimage fw-ConnectIB-MCB194A-
FCA_A1.bin -vpd_r_file ./vpd_r_data.bin
```

2. Disable the Write protection.

```
# flint -d /dev/mst/mt511_pciconf0 -ocr hw set Flash0.WriteProtected=Disabled
```

3. Burn the entire flash.

```
# flint -d /dev/mst/mt5111_pciconf0 -i ./fw-ConnectIB-MCB194A-FCA_A1.bin -ocr -ignore_dev_-
data -allow_psid_change -nofs --yes burn
```

4. Set device manufacture GUIDs.

```
# flint -d /dev/mst/mt511_pciconf0 -ocr -uid 0x0002c903002ef500  smg
```

5. Set device GUIDs.

```
# flint -d /dev/mst/mt511_pciconf0 -ocr -uid 0x0002c903002ef500  sg
```

6. Set Write Protection on the last sector, using the flint 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
```

- ***To view flash settings, run:***

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

- ***To view assigned GUIDs, run:***

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

- ***To change a GUID after the initial burn, run:***

```
# flint -d /dev/mst/mt4113_pciconf0 -ocr -uid 0x0002c903002ef500 sg
```

D.2 ConnectX®-4/ConnectX®-5 Adapter Cards Family

ConnectX-4 adapter cards family consists of ConnectX-4 Lx adapter cards, and ConnectX-5 adapter cards family consists of ConnectX-5 Ex adapter cards.

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/ConnectX®-5 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/ConnectX®-5 adapter cards family for the first time.

For information regarding GUIDs, MACs and VPD, please refer to B.5 “Accessing Remote InfiniBand Device by Direct Route MADs,” on page 171.

D.2.1 Burning the ConnectX®-4/ConnectX®-5 Adapter Cards Family

ConnectX-4 adapter cards family consists of ConnectX-4 Lx adapter cards, and ConnectX-5 adapter cards family consists of ConnectX-5 Ex adapter cards.

Method 1: Generating Firmware with Specific GUIDs and MACs and Burning it on the Device

In order to burn new ConnectX®-4/ConnectX®-5 adapter cards family devices, 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-ConnextX4.mlx -c FW/ MCX454_Ax.ini -wrimage fw-ConnectX4- MCX454_Ax.bin -
base_guid 0xe41d2d0300570fc0-base_mac 0x0000e41d2d570fc0 -vpd_r_file ./vpd_r_data.bin
```

2. Disable the Write protection.

```
# flint -d /dev/mst/mt521_pciconf0 -ocr hw set Flash0.WriteProtected=Disabled
```

3. 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
```

4. Set Write protection

```
# flint -d /dev/mst/mt521_pciconf0 -ocr hw set Flash0.WriteProtected=Top,8-SubSectors
```

5. 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 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-ConnextX4.mlx -c FW/MCX454_Ax.ini -wrimage fw-ConnectX4- MCX454_Ax.bin -vpd_r_file ./vpd_r_data.bin
```

2. Disable the Write protection.

```
# flint -d /dev/mst/mt521_pciconf0 -ocr hw set Flash0.WriteProtected=Disabled
```

3. 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
```

4. Set device manufacture GUIDs and MACs.

```
# flint -d /dev/mst/mt521_pciconf0 -ocr -guid 0xe41d2d0300570fc0 -mac 0x0000e41d2d570fc0 smg
```

5. Set device GUIDs and MACs.

```
# flint -d /dev/mst/mt521_pciconf0 -ocr -guid 0xe41d2d0300570fc0 -mac 0x0000e41d2d570fc0 sg
```

6. Set Write protection on the last sector using flint:

```
# flint -d /dev/mst/mt521_pciconf0 -ocr hw set Flash0.WriteProtected=Top,8-SubSectors
```

7. Enable flash quad SPI IO operations.

```
# flint -d /dev/mst/mt521_pciconf0 -ocr hw set QuadEn=1
```

- ***To view flash settings:***

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

- ***To view assigned GUIDs:***

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

- ***To change a GUID after the initial burn:***

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

- ***To change a MAC after the initial burn:***

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

- ***To change a GUID and derive MAC from it after the initial burn, run:***

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

D.3 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 B.5 “Accessing Remote InfiniBand Device by Direct Route MADs,” on page 171.

D.3.1 Burning the Switch-IB™ Device

The examples below are for managed switches. For unmanaged switches, connect an MTUSB adapter (B.3 “MTUSB-1 USB to I2C Adapter,” on page 167) to the device and use the appropriate mst device (/dev/mst/mtusb…).

Method 1: Generating Firmware with Specific GUIDs and Burning it on the Flash

In order to burn a new Switch-IB™ device, follow the steps below:

1. Generate the initial image with the correct GUIDs and VPD for the specific device using the mlxburn tool. The generated image occupies full flash size.

```
# mlxburn -fw FW/fw-SwitchIB.mlx -c FW/MSB7700-E_Ax.ini -wrimage fw-SwitchIB-MSB7700-E_Ax.bin
-base_guid 0x0002c903002ef500 -vpd_r_file ./vpd_r_data.bin
```

2. Disable the Write protection.

```
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set Flash0.WriteProtected=Disabled
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set Flash1.WriteProtected=Disabled
```

3. Burn the entire flash.

```
# flint -d /dev/mst/mt583_pciconf0 -i ./ fw-SwitchIB-MSB7700-E_Ax.bin -
ocr -ignore_dev_data -allow_psid_change -nofs --yes burn
```

4. Set Write protection.

```
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set Flash0.WriteProtected=Top,2-SubSectors
```

5. Enable flash quad SPI IO operations.

```
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set QuadEn=1
```

Method 2: Generating Firmware Image with Blank GUIDs, Burning and Setting GUIDs on the Flash

In order to burn a new Switch-IB™ device, follow the steps below:

1. Generate the initial image VPD for the specific device using the mlxburn tool. The generated image occupies full flash size.

```
# mlxburn -fw FW/fw-SwitchIB.mlx -c FW/MSB7700-E_Ax.ini -wrimage fw-SwitchIB-MSB7700-E_Ax.bin
-vpd_r_file ./vpd_r_data.bin
```

2. Disable the Write protection.

```
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set Flash0.WritePro-tected=Disabled
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set Flash1.WritePro-tected=Disabled
```

3. Burn the entire flash.

```
# flint -d /dev/mst/mt583_pciconf0 -i ./ fw-SwitchIB-MSB7700-E_Ax.bin -
ocr -ignore_dev_data -allow_psid_change -nofs --yes burn
```

4. Set device manufacture GUIDs.

```
# flint -d /dev/mst/mt583_pciconf0 -ocr -uid 0x0002c903002ef500 smg
```

5. Set device GUIDs.

```
# flint -d /dev/mst/mt583_pciconf0 -ocr -uid 0x0002c903002ef500 sg
```

6. Set Write protection.

```
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set Flash0.WriteProtected=Top,2-SubSectors
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set Flash1.WriteProtected=Top,1-SubSectors
```

7. Enable flash quad SPI IO operations.

```
# flint -d /dev/mst/mt583_pciconf0 -ocr hw set QuadEn=1
```

- ***To view flash settings:***

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

- ***To view assigned GUIDs:***

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

- ***To change a GUID after the initial burn:***

```
# flint -d /dev/mst/mt52000_pciconf0 -ocr -uid 0x0002c903002ef500
sg
```

D.4 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 Appendix B.5,"Accessing Remote InfiniBand Device by Direct Route MADs," on page 171.

D.4.1 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 -wrimage fw-Spectrum-MCB194A-
FCA_A1.bin -base_guid 0x0002c903002ef500 -base_mac 0x02c90ef500 -vpd_r_file ./vpd_r_data.bin
```

2. Disable Write protection.

```
# flint -d /dev/mst/mt585_pciconf0 -override_cache_replacement hw set Flash0.WritePro-
tected=Disabled
```

3. Burn the entire flash.

```
# flint -d /dev/mst/mt585_pciconf0 -i ./fw-Spectrum-MCB194A-FCA_A1.bin -override_cache_re-
placement -ignore_dev_data -nofs -allow_psid_change -y b
```

4. Set Write protection.

```
# flint -d /dev/mst/mt585_pciconf0 -override_cache_replacement hw set
Flash0.WriteProtected=Top,8-SubSectors
```

5. Enable flash quad SPI IO operations.

```
 # flint -d /dev/mst/mt585_pciconf0 -override_cache_replacement hw set QuadEn=1
```

Method 2: Generating a Firmware Image with Blank GUIDs, Burning and Setting GUIDs on the Device

In order to burn a new Spectrum/Switch-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.WritePro-
tected=Disabled
```

3. Burn the entire flash using the flint tool.

```
# flint -d /dev/mst/mt585_pciconf0 -i ./fw-Spectrum-MCB194A-FCA_A1.bin -ocr -ignore_dev_data
-nofs -allow_psid_change -y b
```

4. Set device manufacture GUIDs and MACs.

```
# flint -d /dev/mst/mt585_pciconf0 -ocr -guid 0xe41d2d0300570fc0 -mac 0x0000e41d2d570fc0 smg
```

5. Set device GUIDs and MACs.

```
# flint -d /dev/mst/mt585_pciconf0 -ocr -guid 0xe41d2d0300570fc0 -mac 0x0000e41d2d570fc0 sg
```

6. Set Write protection on the last sector.

```
# flint -d /dev/mst/mt585_pciconf0 -ocr hw set Flash0.WriteProtected=Top,8-SubSectors
```

7. Enable flash quad SPI IO operations.

```
 # flint -d /dev/mst/mt585_pciconf0 -ocr hw set QuadEn=1
```

- ***To view flash settings:***

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

- ***To view assigned GUIDs:***

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

- ***To change a GUID after the initial burn:***

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

- ***To change a MAC after the initial burn:***

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

- ***To change a GUID and derive MAC from it after the initial burn, run:***

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

Appendix E: Generating Firmware Secure and NV LifeCycle Configurations Files

E.1 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 2.3.10, 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 follow:

   ```
   <forbidden_fw_version index='1..31' >0x0</forbidden_fw_version>
   ```

5. Run the xml2bin command to generate the binary file.

E.2 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 2.3.13).