Software Installation and Upgrade

Table of contents

Info

It is recommended to upgrade your BlueField product to the latest software and firmware versions available to benefit from new features and latest bug fixes.

NVIDIA® BlueField® is shipped with the BlueField software based on Ubuntu 22.04 pre-installed. The DPU's Arm execution environment has the capability of being functionally isolated from the host server and uses a dedicated network management interface (separate from the host server's management interface). The Arm cores can run the Open vSwitch Database (OVSDB) or other virtual switches to create a secure solution for bare metal provisioning.

The software package also includes support for DPDK as well as applications for accelerated encryption.

The BlueField DPU supports several methods for OS deployment and upgrade:

- Full OS image deployment using a BlueField boot stream file (BFB) via RShim interface

  Info

  This installation method is compatible with SuperNICs.

- Full OS deployment using PXE which can be used over different network interfaces available on the BlueField DPU (1GbE mgmt, tmfifo or NVIDIA® ConnectX®)
- Individual packages can be installed or upgraded using standard Linux package management tools (e.g., apt, dpkg, etc.)

The DPU's BMC software (i.e., BMC firmware, ERoT firmware, DPU golden image, and NIC firmware golden image) is included in the BF-Bundle and BF-FW-Bundle BFB files. The BFB installation updates BMC software components automatically if BMC credentials (i.e., `BMC_USER` and `BMC_PASSWORD`) are provided in bf.cfg.

Info

The minimum BMC Firmware version that supports this method of BMC upgrade from BFB image, is 23.07. If your BMC firmware version is lower, follow the *NVIDIA BlueField BMC Software* documentation to upgrade BMC firmware. The BMC version can be obtained by following instructions here.

Info

`BMC_REBOOT="no"` by default. This will require BMC to be rebooted after BFB installation process finish to apply the new BMC firmware version. BMC reboot will reset the BMC console.

Info

Upgrading BlueField software using BFB-bundle, performs NIC firmware update by default. Host system-level reset or power cycle is required to apply the new firmware.

A firmware-only BFB, `bf-fwbundle-<version>.prod.bfb`, is available for BlueField devices for day 2 operations, updating all firmware components aside from Arm OS and DOCA software.

The firmware-only BFB can use the same set of bf.cfg parameters as a standard BFB with the exception of BlueField-OS related flags (e.g., `UPDATE_DPU_OS`).

Types and Methods of Updating BlueField Software Image

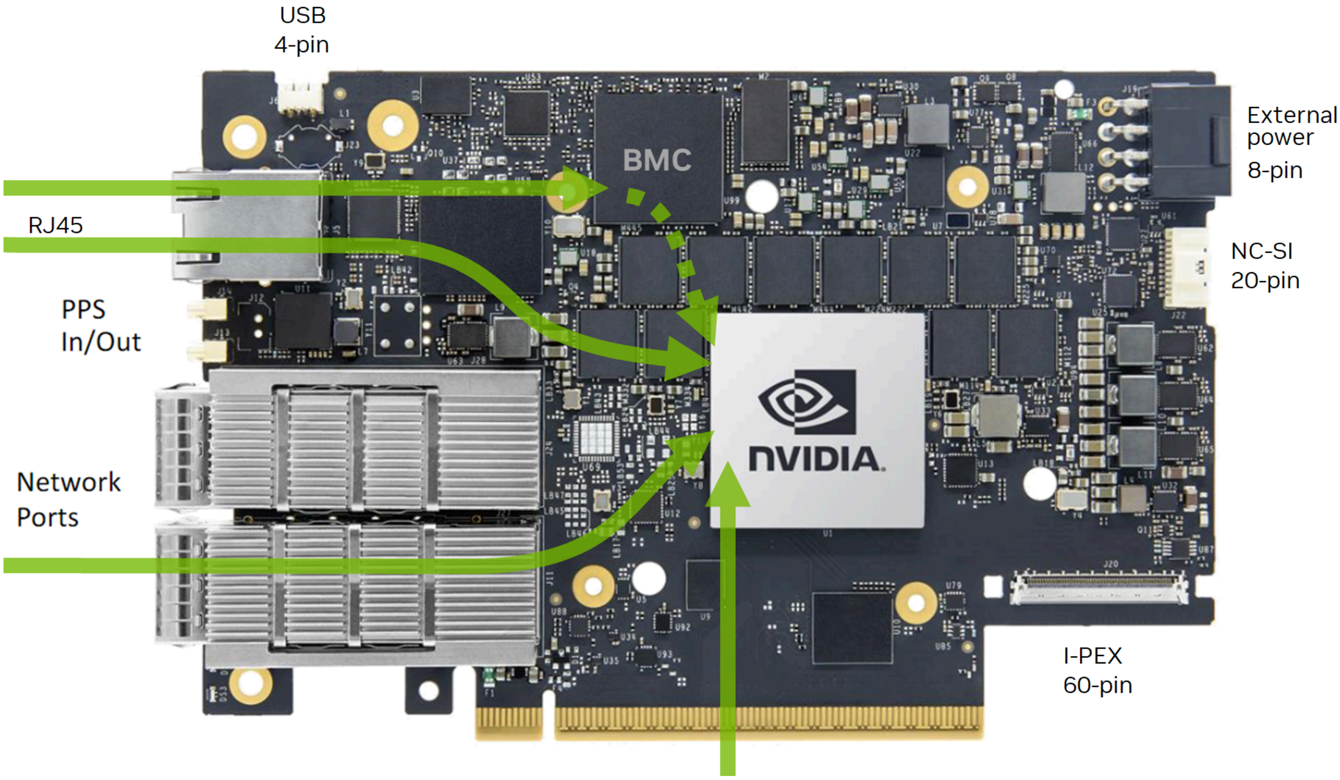
Interfaces and Methods

NVIDIA® BlueField® has multiple paths to update the running software depending on the BlueField interface used:

- RJ-45 1GbE interface:
    - BlueField BMC using Redfish – Recommended for managing BlueField devices at scale in a data center. See the *BlueField Management and Initial Provisioning* and the "Update and Recovery" section of the *NVIDIA BlueField BMC Software* documentation for detailed instructions.
    - BlueField Arm (DPU mode only) –
        - PXE/HTTP boot – BlueField, like a standard server, incorporates a UEFI BIOS that can be used to PXE/HTTP boot the device and pull a new image. See "Deploying BlueField Software Using PXE" for more information.
        - Linux standard tools – done by running `apt update`/`yum install` from within the Arm OS running on BlueField. See "Upgrading BlueField Using Standard Linux Tools" for more information.
- High-speed network ports:
    - BlueField Arm (DPU mode only) – This option is the same as the BlueField Arm scenario above, but it uses the high-speed network ports instead of the RJ-45 1GbE interface.
        - PXE/HTTP boot – BlueField's UEFI BIOS may be used to PXE/HTTP boot the device and pull a new image. See "Deploying BlueField Software Using PXE" for more information .

        - Linux standard tools – done by running `apt update` / `yum install` from within the Arm OS running on BlueField. See "Upgrading BlueField Using Standard Linux Tools" for more information.

- Host server's PCIe interface:
    - From server host-OS – Using the BlueField Management PCIe RShim PF interface. See "Deploying BlueField Software Using BFB from Host" for more details.
    - From the server's platform-BMC – PLDM firmware update (PLDM over MCTP over PCIe), a standardized protocol that enables out-of-band firmware upgrades for devices via the server's platform BMC (does not include the DPU OS).



BlueField Image Types

BlueField software images are available in the following formats:

- BFB – BlueField Bundle images – a proprietary format used to update and recover the BlueField device. There are two types of BFB images available:
    - BF-Bundle – Includes BlueField firmware components, Arm OS, and DOCA. This bundle contains everything needed to update all possible components of a

  BlueField device.
  - BF-FW-Bundle – Includes only the BlueField firmware components and excludes Arm OS and DOCA. This is typically used for day-2 operations or by customers working in NIC mode who do not require Arm OS or DOCA running on the device.
- ISO – Similar to BF-Bundle, this format includes BlueField firmware components, Arm OS, and DOCA packages in an ISO standard format. The BlueField ISO image is based on the standard Ubuntu ISO image for Arm64, but with an updated kernel and added DOCA packages. PXE booting the BlueField device with the ISO image results in the installation of the Arm OS, including DOCA, and the update of BlueField firmware components.
- PLDM – Similar to BF-FW-Bundle, this format includes only BlueField firmware components and does not include Arm OS or DOCA. The image is distributed per SKU to keep the image size small, as required by the limitations of some platform BMCs.
- Repository – An online repository used for updating with standard Linux tools. This method updates DOCA from NVIDIA's repository and Arm OS packages from Canonical's updates repository. BlueField firmware components are also updated. Post-installation steps are required to upgrade BlueField firmware components (i.e., ATF/UEFI, NIC firmware, and BMC components). See "Updating BlueField Software Packages Using Standard Linux Tools" for details.

Note

Make sure that the BlueField firmware versions are aligned to the same release with the running DOCA version on the BlueField Arm.

The following table summarizes image types, their contents, and from which interfaces they can be loaded:

Interface	Method	Image Type	BlueField Operation Modes which Support this Flow	Includes ATF, UEFI, BMC, CEC, NIC-FW	Includes Arm-OS	Includes DOCA	Normal Service operation during update flow	Notes
RJ-45 1GbE	BlueField BMC using Redfish	BF-Bundle – bfb	- NIC mode - DPU mode	Yes	Yes	Yes	No	Server power cycle [1] is required to apply configuration.
		BF-FW-Bundle – bfb	- NIC mode - DPU mode	Yes	No	No	No	Server power cycle [1] is required to apply configuration.
	BlueField Arm – PXE/HTTP boot	BF-Bundle – bfb	DPU mode	Yes	Yes	Yes	No	User should convert the original BFB file to a bootable file. See "Deploying BlueField Software Using BFB with PXE" for details. Server power cycle [1] is required to apply configuration.
		BF-FW-Bundle – bfb	DPU mode	Yes	No	No	No	User should convert the original BFB file to a bootable file. See "Deploying BlueField Software Using BFB with PXE" for details. Server power cycle [1] is required to apply configuration.
		ISO	DPU mode	Yes	Yes	Yes	No	Server power cycle [1] is required to

Interface	Method	Image Type	BlueField Operation Modes which Support this Flow	Includes ATF, UEFI, BMC, CEC, NIC-FW	Includes Arm-OS	Includes DOCA	Normal Service operation during update flow	Notes
								apply configuration.
	Linux Standard tools	apt/yum update	DPU mode	Yes	No	Yes	No	Limitations exist. Server power cycle [1] is required to apply configuration.
High speed network ports	BlueField Arm - PXE/HTTP boot	BF-Bundle – bfb	DPU mode	Yes	Yes	Yes	No	User should convert the original BFB file to a bootable file. See "Deploying BlueField Software Using BFB with PXE" for details. Server power cycle [1] is required to apply configuration.
		BF-FW-Bundle – bfb	DPU mode	Yes	No	No	No	User should convert the original BFB file to a bootable file. See "Deploying BlueField Software Using BFB with PXE" for details. Server power cycle [1] is required to apply configuration.
		ISO	DPU mode	Yes	Yes	Yes	No	Server power cycle [1] is required to apply configuration.
	Linux Standard tools	apt/yum update	DPU mode	Yes	No	Yes	No	Limitations exist.

Interface	Method	Image Type	BlueField Operation Modes which Support this Flow	Includes ATF, UEFI, BMC, CEC, NIC-FW	Includes Arm-OS	Includes DOCA	Normal Service operation during update flow	Notes
								Server power cycle [1] is required to apply configuration.
Host server's PCIe	From server host-OS	BF-Bundle – bfb	• NIC mode • DPU mode	Yes	Yes	Yes	No	Server power cycle [1] is required to apply configuration.
		BF-FW-Bundle – bfb	• NIC mode • DPU mode	Yes	No	No	No	Server power cycle [1] is required to apply configuration.
	From server's platform BMC	PLDM image	• NIC mode • DPU mode	Yes	No	No	Yes – no impact on service during update, until next restart of device	Apply on next server power cycle [1]

1. BlueField must be restarted to apply changes. When in DPU mode (Arm OS is used), the running Arm OS must be gracefully shutdown prior to power cycle. Instead of a power cycle, server reboot may be performed provided a graceful shutdown of the Arm OS and a manual issue of reset/restart of BlueField BMC using Redfish are performed.

Where to Download BlueField Software Images

Go to NVIDIA DOCA Download page and download the appropriate image.

Deploying BlueField Software Using BFB from Host

Info

It is recommended to upgrade your BlueField product to the latest software and firmware versions available to benefit from new features and latest bug fixes.

Note

This procedure assumes that a NVIDIA® BlueField® networking platform (DPU or SuperNIC) has already been installed in a server according to the instructions detailed in the BlueField device's hardware user guide.

The following table lists an overview of the steps required to install Ubuntu BFB on your BlueField:

Step	Procedure	Link to Section
1	Uninstall previous DOCA on host (if exists)	Uninstall Previous Software from Host
2	Install RShim on the host	Install RShim on Host
3	Verify that RShim is running on the host	Ensure RShim Running on Host
4	Install the Ubuntu BFB image	BFB Installation
5	Verify installation completed successfully	Verify BFB is Installed

Prepare Host for BFB Update Flow

Uninstall Previous Software from Host

If an older DOCA software version is installed on your host, make sure to uninstall it before proceeding with the installation of the new version:

Ubuntu

```
host# for f in $( dpkg --list | grep doca | awk '{print $2}' ); do echo $f ; apt remove --purge $f -y ; done
host# sudo apt-get autoremove
```

CentOS/RHEL

```
host# for f in $(rpm -qa |grep -i doca ) ; do yum -y remove $f; done
host# yum autoremove
host# yum makecache
```

Install RShim on Host

Before installing the RShim driver, verify that the RShim devices, which will be probed by the driver, are listed under `lsusb` or `lspci`.

```
lspci | grep -i nox
```

Output example:

```
27:00.0 Ethernet controller: Mellanox Technologies MT42822 BlueField-2 integrated ConnectX-6 Dx network controller
27:00.1 Ethernet controller: Mellanox Technologies MT42822 BlueField-2 integrated ConnectX-6 Dx network controller
```

```
27:00.2 Non-Volatile memory controller: Mellanox Technologies NVMe SNAP Controller
27:00.3 DMA controller: Mellanox Technologies MT42822 BlueField-2 SoC Management Interface // This is the RShim PF
```

RShim is compiled as part of the `doca-runtime` package in the `doca-host-repo-ubuntu<version>_amd64` file (`.deb` or `.rpm`).

To install `doca-runtime`:

OS	Procedure

Ubuntu/Debian	1. Download the DOCA Runtime host package from the "Installation Files" section in the *NVIDIA DOCA Installation Guide for Linux*.
	2. Unpack the deb repo. Run:
	`host# sudo dpkg -i doca-host-repo-ubuntu<version>_amd64.deb`
	3. Perform apt update. Run:
	`host# sudo apt-get update`
	4. Run `apt install` for DOCA runtime package.
	`host# sudo apt install doca-runtime`

CentOS/RHEL 7.x	1. Download the DOCA runtime host package from the "Installation Files" section in the *NVIDIA DOCA Installation Guide for Linux*. 2. Unpack the RPM repo. Run: `host# sudo rpm -Uvh doca-host-repo-rhel<version>.x86_64.rpm` 3. Enable new yum repos. Run:

OS	Procedure

	`host# sudo yum makecache`
	4. Run `yum install` to install DOCA runtime package.
	`host# sudo yum install doca-runtime`

CentOS/RHEL 8.x or Rocky 8.6	1. Download the DOCA runtime host package from the "Installation Files" section in the *NVIDIA DOCA Installation Guide for Linux*.
	2. Unpack the RPM repo. Run:
	`host# sudo rpm -Uvh doca-host-repo-rhel<version>.x86_64.rpm`
	3. Enable new dnf repos. Run:
	`host# sudo dnf makecache`
	4. Run `dnf install` to install DOCA runtime.
	`host# sudo dnf install doca-runtime`

Ensure RShim Running on Host

1. Verify RShim status. Run:

```
sudo systemctl status rshim
```

Expected output:

```
active (running)
...
Probing pcie-0000:<BlueField's PCIe Bus address on host>
create rshim pcie-0000:<BlueField's PCIe Bus address on host>
rshim<N> attached
```

Where `<N>` denotes RShim enumeration starting with 0 (then 1, 2, etc.) for every additional BlueField installed on the server.

If the text `another backend already attached` is displayed, users will not be able to use RShim on the host.

1. If the previous command displays `inactive` or another error, restart RShim service. Run:

   ```
   sudo systemctl restart rshim
   ```

2. Verify RShim status again. Run:

   ```
   sudo systemctl status rshim
   ```

2. Display the current setting. Run:

   ```
   # cat /dev/rshim<N>/misc | grep DEV_NAME
   DEV_NAME        pcie-0000:04:00.2
   ```

   This output indicates that the RShim service is ready to use.

Updating BlueField Software using BFB image

BFB Image Types

The BFB image is available in two formats:

- BF-Bundle – includes BlueField firmware and BlueField Arm OS as well as DOCA
- BF-FW-Bundle – includes BlueField firmware only

Select the appropriate image for you.

Info

Both images end with `*.bfb`.

Downloading the BFB Image

To download the BFB image, BF-Bundle or BF-FW-Bundle go to the NVIDIA DOCA Downloads page.

BFB Installation

Info

To upgrade the BMC firmware using BFB, the user must provide the current BMC credentials in the `bf.cfg`. Refer to "Customizing BlueField Software Deployment" for more information.

Note

Upgrading the BlueField networking platform using BFB Bundle updates the NIC firmware by default. NIC firmware upgrade triggers a NIC reset flow via `mlxfwreset` in the BlueField Arm.

If this reset flow cannot complete or is not supported on your setup, `bfb-install` alerts about it at the end of the installation. In this case, perform a BlueField system-level reset.

To skip NIC firmware upgrade during BFB Bundle installation , provide the parameter `WITH_NIC_FW_UPDATE=no` in the `bf.cfg` text file when running `bfb-install` .

Note

All new BlueField-2 devices and all BlueField-3 are secure boot enabled, hence all the relevant SW images (ATF/UEFI, Linux Kernel and Drivers) must be signed in order to boot. All formally published SW images are signed.

Warning

When installing the BFB bundle in NIC mode, users must perform the following:

1. Prior to installing the BFB bundle, users must unbind each NIC port, using its PCIe function address. For example:

```
[host]# lspci -d 15b3:
21:00.0 Ethernet controller: Mellanox
Technologies MT43244 BlueField-3 integrated
ConnectX-7 network controller (rev 01)
21:00.1 Ethernet controller: Mellanox
Technologies MT43244 BlueField-3 integrated
```

```
ConnectX-7 network controller (rev 01)
21:00.2 DMA controller: Mellanox Technologies
MT43244 BlueField-3 SoC Management Interface
(rev 01)

[host]# echo 0000:21:00.0 >
/sys/bus/pci/drivers/mlx5_core/unbind
[host]# echo 0000:21:00.1 >
/sys/bus/pci/drivers/mlx5_core/unbind
```

If there are multiple BlueField devices to be updated in the server, repeat this step on all of them, before starting BFB bundle installations.

2. After the BFB bundle installation is done, users must perform a warm reboot or power cycle on the host.

To install the BFB image, run on the host side:

```
# bfb-install -h
syntax: bfb-install --bfb|-b <BFBFILE> [--config|-c <bf.cfg>] \
  [--rootfs|-f <rootfs.tar.xz>] --rshim|-r <rshimN> [--help|-h]
```

The `bfb-install` utility is installed by the RShim package.

This utility script pushes the BFB image and optional configuration (`bf.cfg` file) to the BlueField side and checks and prints the BFB installation progress. To see the BFB installation progress, please install the `pv` Linux tool.

⚠ **Warning**

BFB image installation must complete before restarting the system/BlueField. Doing so may result in anomalous behavior of the

BlueField (e.g., it may not be accessible using SSH). If this happens, re-initiate the update process with `bfb-install` to recover the BlueField.

The following is an output example of the installation of a BF-Bundle (including BlueField Arm OS Ubuntu 22.04) with the `bfb-install` script assuming `pv` has been installed:

```
# bfb-install --bfb <BlueField-BSP>.bfb --config bf.cfg --rshim
rshim0 Pushing bfb + cfg
1.46GiB 0:01:11 [20.9MiB/s] [
<=>                        ]
Collecting BlueField booting status. Press Ctrl+C to stop…
 INFO[PSC]: PSC BL1 START
 INFO[BL2]: start
 INFO[BL2]: boot mode (rshim)
 INFO[BL2]: VDDQ: 1120 mV
 INFO[BL2]: DDR POST passed
 INFO[BL2]: UEFI loaded
 INFO[BL31]: start
 INFO[BL31]: lifecycle Production
 INFO[BL31]: MB8: VDD adjustment complete
 INFO[BL31]: VDD: 743 mV
 INFO[BL31]: power capping disabled
 INFO[BL31]: runtime
 INFO[UEFI]: eMMC init
 INFO[UEFI]: eMMC probed
 INFO[UEFI]: UPVS valid
 INFO[UEFI]: PMI: updates started
 INFO[UEFI]: PMI: total updates: 1
 INFO[UEFI]: PMI: updates completed, status 0
 INFO[UEFI]: PCIe enum start
 INFO[UEFI]: PCIe enum end
 INFO[UEFI]: UEFI Secure Boot (disabled)
 INFO[UEFI]: exit Boot Service
```

```
INFO[MISC]: : Found bf.cfg
INFO[MISC]: : Ubuntu installation started
INFO[MISC]: bfb_pre_install
INFO[MISC]: Installing OS image
INFO[MISC]: : Changing the default password for user ubuntu
INFO[MISC]: : Running bfb_modify_os from bf.cfg
INFO[MISC]: : Ubuntu installation finished
```

Verify BFB is Install Completed Successfully

In DPU mode, after installation of the Ubuntu OS is complete, the following note appears in `/dev/rshim0/misc` on first boot:

```
...
INFO[MISC]: Linux up
INFO[MISC]: DPU is ready
```

`DPU is ready` indicates that all the relevant services are up, and users can log into the system.

After the installation of the Ubuntu 22.04 BFB, the configuration detailed in the following sections is generated.

Note

Make sure all the services (including cloud-init) are started on BlueField and to perform a graceful shutdown before power cycling the host server.

BlueField OS image version is stored under `/etc/mlnx-release` in the BlueField:

```
# cat /etc/mlnx-release
bf-bundle-2.9.0-<version>_ubuntu-22.04_prod
```

Check the NIC firmware version from the host and make sure the new version is applied:

```
# flint -d /dev/mst/mt41692_pciconf0 q
Image type:            FS4
FW Version:            32.43.0366
FW Version(Running):   32.43.0318
FW Release Date:       12.10.2024
Product Version:       32.43.0318
Rom Info:              type=UEFI Virtio net version=21.4.13
cpu=AMD64,AARCH64
                       type=UEFI Virtio blk version=22.4.14
cpu=AMD64,AARCH64
                       type=UEFI version=14.36.12
cpu=AMD64,AARCH64
                       type=PXE version=3.7.500 cpu=AMD64
Description:           UID                GuidsNumber
Base GUID:             c470bd0300cbe708        38
Base MAC:              c470bdcbe708            38
Image VSD:             N/A
Device VSD:            N/A
PSID:                  MT_0000000001
Security Attributes:   secure-fw
```

If the version of the NIC firmware is different from the running firmware version as is the case in this example, then a BlueField system-level reset is required.

ⓘ **Info**

To verify the version of the installed BMC components, refer to the BMC documentation:

- Retrieving Golden Image Version Information Using Redfish
- Fetching Running BMC Firmware Version
- Fetching Running CEC Firmware Version

In NIC mode, verify the NIC firmware and BMC components versions using Redfish.

Apply New BFB Image

BlueField must be restarted to apply the new firmware. To restart BlueField:

1. Perform a graceful shutdown of the BlueField Arm OS.
2. Power cycle the server to complete the restart.

Alternatively, a server reboot may be done instead of power cycle by following these steps:

1. Graceful shutdown the BlueField Arm OS.

   ⓘ **Info**

   Without graceful shutdown of BlueField Arm OS during server reboot, the BlueField Arm side does not undergo a restart process (so only NIC firmware is applied).

2. Wait until completed.
3. Reboot the server (ATF, UEFI, BlueField Arm OS, NIC firmware is applied).

Info

Server reboot will not restart the BlueField BMC (CEC not applied).

4. Log into BlueField BMC via Redfish and issue a restart (BlueField BMC and CEC is applied).

Optional Steps Post Installation of BF-Bundle (DPU Mode with BlueField Arm OS Running)

Updating NVConfig Params from Host

1. Optional. To reset the BlueField NIC firmware configuration (aka Nvconfig params) to their factory default values, run the following from the BlueField ARM OS or from the host OS:

```
# sudo mlxconfig -d /dev/mst/<MST device> -y reset

Reset configuration for device /dev/mst/<MST device>? (y/n)
[n] : y
Applying... Done!
-I- Please reboot machine to load new configurations.
```

Note

For now, please ignore tool's instruction to reboot

Note

To learn what MST device the BlueField has on your setup, run:

```
mst start
mst status
```

Example output taken on a multiple BlueField host:

```
// The MST device corresponds with PCI Bus
address.

MST modules:
------------
    MST PCI module is not loaded
    MST PCI configuration module loaded

MST devices:
------------
/dev/mst/mt41692_pciconf0        - PCI
configuration cycles access.

domain:bus:dev.fn=0000:03:00.0 addr.reg=88
data.reg=92 cr_bar.gw_offset=-1
                                   Chip
revision is: 01
/dev/mst/mt41692_pciconf1        - PCI
configuration cycles access.

domain:bus:dev.fn=0000:83:00.0 addr.reg=88
data.reg=92 cr_bar.gw_offset=-1
                                   Chip
revision is: 01
```

```
/dev/mst/mt41686_pciconf0        - PCI
configuration cycles access.

domain:bus:dev.fn=0000:a3:00.0 addr.reg=88
data.reg=92 cr_bar.gw_offset=-1
                                   Chip
revision is: 01
```

The MST device IDs for the BlueField-2 and BlueField-3 devices in this example are `/dev/mst/mt41686_pciconf0` and `/dev/mst/mt41692_pciconf0` respectively.

2. (Optional) Enable NVMe emulation. Run:

```
sudo mlxconfig -d <MST device> -y s NVME_EMULATION_ENABLE=1
```

3. Skip this step if your BlueField is Ethernet only. Please refer to section "Supported Platforms and Interoperability" under the Release Notes to learn your BlueField type.

   If you have an InfiniBand-and-Ethernet-capable BlueField, the default link type of the ports will be configured to IB. If you want to change the link type to Ethernet, please run the following configuration:

```
sudo mlxconfig -d <MST device> -y s LINK_TYPE_P1=2
LINK_TYPE_P2=2
```

4. Perform a BlueField system-level reset for the new settings to take effect.

Note

After modifying files on the BlueField, run the command `sync` to flush file system buffers to eMMC/SSD flash memory to avoid data loss during reboot or power cycle.

Default Network Interface Configuration

Network interfaces are configured using the `netplan` utility:

```
# cat /etc/netplan/50-cloud-init.yaml
# This file is generated from information provided by the datasource.  Changes
# to it will not persist across an instance reboot.  To disable cloud-init's
# network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
    ethernets:
        tmfifo_net0:
            addresses:
            - 192.168.100.2/30
            dhcp4: false
            nameservers:
                addresses:
                - 192.168.100.1
            routes:
            -   metric: 1025
                to: 0.0.0.0/0
                via: 192.168.100.1
        oob_net0:
            dhcp4: true
    renderer: NetworkManager
    version: 2
```

```
# cat /etc/netplan/60-mlnx.yaml
network:
  ethernets:
    enp3s0f0s0:
      dhcp4: 'true'
    enp3s0f1s0:
      dhcp4: 'true'
  renderer: networkd
  version: 2
```

BlueField devices also have a local IPv6 (LLv6) derived from the MAC address via the STD stack mechanism. For a default MAC, `00:1A:CA:FF:FF:01`, the LLv6 address would be `fe80::21a:caff:feff:ff01`.

For multi-device support, the LLv6 address works with SSH for any number of BlueField devices in the same host by including the interface name in the SSH command:

```
host]# systemctl restart rshim
// wait 10 seconds
host]# ssh -6 ubuntu@fe80::21a:caff:feff:ff01%tmfifo_net<n>
```

> **Note**
>
> If `tmfifo_net<n>` on the host does not have an LLv6 address, restart the RShim driver:
>
> ```
> systemctl restart rshim
> ```

Default Ports and OVS Configuration

The `/sbin/mlnx_bf_configure` script runs automatically with `ib_umad` kernel module loaded (see `/etc/modprobe.d/mlnx-bf.conf`) and performs the following configurations:

1. Ports are configured with switchdev mode and software steering.
2. RDMA device isolation in network namespace is enabled.
3. Two scalable function (SF) interfaces are created (one per port) if BlueField is configured with Embedded CPU mode (default):

```
# mlnx-sf -a show

SF Index: pci/0000:03:00.0/229408
  Parent PCI dev: 0000:03:00.0
  Representor netdev: en3f0pf0sf0
  Function HWADDR: 02:a9:49:7e:34:29
  Function trust: off
  Function roce: true
  Function eswitch: NA
  Auxiliary device: mlx5_core.sf.2
    netdev: enp3s0f0s0
    RDMA dev: mlx5_2

SF Index: pci/0000:03:00.1/294944
  Parent PCI dev: 0000:03:00.1
  Representor netdev: en3f1pf1sf0
  Function HWADDR: 02:53:8f:2c:8a:76
  Function trust: off
  Function roce: true
  Function eswitch: NA
  Auxiliary device: mlx5_core.sf.3
    netdev: enp3s0f1s0
    RDMA dev: mlx5_3
```

The parameters for these SFs are defined in configuration file `/etc/mellanox/mlnx-sf.conf`.

```
/sbin/mlnx-sf --action create --device 0000:03:00.0 --sfnum 0
--hwaddr 02:61:f6:21:32:8c
/sbin/mlnx-sf --action create --device 0000:03:00.1 --sfnum 0
--hwaddr 02:30:13:6a:2d:2c
```

ⓘ **Note**

To avoid repeating a MAC address in the your network, the SF MAC address is set randomly upon BFB installation. You may choose to configure a different MAC address that better suit your network needs.

4. Two OVS bridges are created:

```
# ovs-vsctl show
f08652a8-92bf-4000-ba0b-7996c772aff6
    Bridge ovsbr2
        Port ovsbr2
            Interface ovsbr2
                type: internal
        Port p1
            Interface p1
        Port en3f1pf1sf0
            Interface en3f1pf1sf0
        Port pf1hpf
            Interface pf1hpf
    Bridge ovsbr1
        Port p0
            Interface p0
        Port pf0hpf
            Interface pf0hpf
        Port ovsbr1
            Interface ovsbr1
                type: internal
        Port en3f0pf0sf0
            Interface en3f0pf0sf0
    ovs_version: "2.14.1"
```

The parameters for these bridges are defined in configuration file `/etc/mellanox/mlnx-ovs.conf`:

```

CREATE_OVS_BRIDGES="yes"
OVS_BRIDGE1="ovsbr1"
OVS_BRIDGE1_PORTS="p0 pf0hpf en3f0pf0sf0"
OVS_BRIDGE2="ovsbr2"
OVS_BRIDGE2_PORTS="p1 pf1hpf en3f1pf1sf0"
OVS_HW_OFFLOAD="yes"
OVS_START_TIMEOUT=30
```

Note

If failures occur in `/sbin/mlnx_bf_configure` or configuration changes happen (e.g. switching to separated host mode) OVS bridges are not created even if `CREATE_OVS_BRIDGES="yes"`.

5. OVS HW offload is configured.

DHCP Client Configuration

```
/etc/dhcp/dhclient.conf:
send vendor-class-identifier "NVIDIA/BF/DP";
interface "oob_net0" {
  send vendor-class-identifier "NVIDIA/BF/OOB";
        }
```

Ubuntu Boot Time Optimizations

To improve the boot time, the following optimizations were made to Ubuntu OS image:

```
# cat /etc/systemd/system/systemd-networkd-wait-online.service.d/override.conf
[Service]
ExecStart=
ExecStart=/usr/bin/nm-online -s -q --timeout=5

# cat /etc/systemd/system/NetworkManager-wait-online.service.d/override.conf
[Service]
ExecStart=
ExecStart=/usr/lib/systemd/systemd-networkd-wait-online --timeout=5

# cat /etc/systemd/system/networking.service.d/override.conf
[Service]
TimeoutStartSec=5
ExecStop=
ExecStop=/sbin/ifdown -a --read-environment --exclude=lo --force
--ignore-errors
```

This configuration may affect network interface configuration if DHCP is used. If a network device fails to get configuration from the DHCP server, then the timeout value in the two files above must be increased.

Grub Configuration:

Setting the Grub timeout at 2 seconds with `GRUB_TIMEOUT=2` under `/etc/default/grub`. In conjunction with the `GRUB_TIMEOUT_STYLE=countdown` parameter, Grub will show the countdown of 2 seconds in the console before booting Ubuntu. Please note that, with this short timeout, the standard Grub method for entering the Grub menu (i.e., SHIFT or Esc) does not work. Function key F4 can be used to enter the Grub menu.

System Services:

docker.service is disabled in the default Ubuntu OS image as it dramatically affects boot time.

The `kexec` utility can be used to reduce the reboot time. Script `/usr/sbin/kexec_reboot` is included in the default Ubuntu 22.04 OS image to run corresponding `kexec` commands.

```
# kexec_reboot
```

Ubuntu Dual Boot Support

BlueField may be installed with support for dual boot. That is, two identical images of the BlueField OS may be installed using BFB.

The following is a proposed SSD partitioning layout for 119.24 GB SSD:

```
Device                  Start       End   Sectors  Size Type
```

```
/dev/nvme0n1p1       2048    104447    102400   50M EFI System
/dev/nvme0n1p2     104448 114550086 114445639 54.6G Linux
filesystem
/dev/nvme0n1p3 114550087 114652486    102400   50M EFI System
/dev/nvme0n1p4 114652487 229098125 114445639 54.6G Linux
filesystem
/dev/nvme0n1p5 229098126 250069645  20971520   10G Linux
filesystem
```

Where:

- `/dev/nvme0n1p1` – boot EFI partition for the first OS image
- `/dev/nvme0n1p2` – root FS partition for the first OS image
- `/dev/nvme0n1p3` – boot EFI partition for the second OS image
- `/dev/nvme0n1p4` – root FS partition for the second OS image
- `/dev/nvme0n1p5` – common partition for both OS images

For example, the following is a proposed eMMC partitioning layout for a 64GB eMMC:

```

Device             Start       End  Sectors  Size Type
/dev/mmcblk0p1      2048    104447   102400   50M EFI System
/dev/mmcblk0p2    104448  50660334 50555887 24.1G Linux
filesystem
/dev/mmcblk0p3  50660335  50762734   102400   50M EFI System
/dev/mmcblk0p4  50762735 101318621 50555887 24.1G Linux
filesystem
/dev/mmcblk0p5 101318622 122290141 20971520   10G Linux
filesystem
```

Where:

- `/dev/mmcblk0p1` – boot EFI partition for the first OS image
- `/dev/mmcblk0p2` – root FS partition for the first OS image
- `/dev/mmcblk0p3` – boot EFI partition for the second OS image
- `/dev/mmcblk0p4` – root FS partition for the second OS image
- `/dev/mmcblk0p5` – common partition for both OS images

> ⓘ **Note**
>
> The common partition can be used to store BFB files that will be used for OS image update on the non-active OS partition.

Installing Ubuntu OS Image Using Dual Boot

> ⓘ **Note**
>
> For software upgrade procedure, please refer to section "Upgrading Ubuntu OS Image Using Dual Boot".

Add the values below to the `bf.cfg` configuration file (see section "bf.cfg Parameters" for more information).

```
DUAL_BOOT=yes
```

If the eMMC size is ≤16GB, dual boot support is disabled by default, but it can be forced by setting the following parameter in `bf.cfg`:

```
FORCE_DUAL_BOOT=yes
```

To modify the default size of the `/common` partition, add the following parameter:

```
COMMON_SIZE_SECTORS=<number-of-sectors>
```

The number of sectors is the size in bytes divided by the block size (512). For example, for 10GB, the `COMMON_SIZE_SECTORS=$((10*2**30/512))`.

After assigning size for the `/common` partition, what remains is divided equally between the two OS images.

```
# bfb-install --bfb <BFB> --config bf.cfg --rshim rshim0
```

This will result in the Ubuntu OS image to be installed twice on the BlueField.

> **Note**
>
> For comprehensive list of the supported parameters to customize `bf.cfg` during BFB installation, refer to section "bf.cfg Parameters".

Upgrading Ubuntu OS Image Using Dual Boot

1. Download the new BFB to the BlueField into the `/common` partition. Use `bfb_tool.py` script to install the new BFB on the inactive BlueField partition:

   ```
   /opt/mellanox/mlnx_snap/exec_files/bfb_tool.py --op
   ```

```
fw_activate_bfb --bfb <BFB>
```

2. Reset BlueField to load the new OS image:

   ```
   /sbin/shutdown -r 0
   ```

   BlueField should now boot into the new OS image.

Use `efibootmgr` utility to manage the boot order if necessary.

- Change the boot order with:

  ```
  # efibootmgr -o
  ```

- Remove stale boot entries with:

  ```
  # efibootmgr -b <E> -B
  ```

  Where `<E>` is the last character of the boot entry (i.e., `Boot000<E>`). You can find that by running:

  ```
  # efibootmgr
  BootCurrent: 0040
  Timeout: 3 seconds
  BootOrder: 0040,0000,0001,0002,0003
  Boot0000* NET-NIC_P0-IPV4
  Boot0001* NET-NIC_P0-IPV6
  Boot0002* NET-NIC_P1-IPV4
  Boot0003* NET-NIC_P1-IPV6
  Boot0040* focal0
  ```

```
....2
```

Note

Modifying the boot order with `efibootmgr -o` does not remove unused boot options. For example, changing a boot order from 0001,0002, 0003 to just 0001 does not actually remove 0002 and 0003. 0002 and 0003 need to be explicitly removed using `efibootmgr -B`.

Deploying BlueField Software from BlueField BMC

For information on deploying BlueField software using BFB from BlueField BMC, please refer to the *NVIDIA BlueField BMC Software User Manual*.

Deploying BlueField Software Using PXE

The BlueField boot flow is similar to a standard server, allowing to boot via an external PXE/HTTP server.

Providing a BFB or ISO image via PXE server enables updating the BlueField software.

The update flow for the two software update image types, BFB and ISO, is slightly different as elaborated on in the following subpages.

BFB Update vs. ISO Update

- PXE booting the BlueField device with a BFB file updates all components included in the BFB image at the end of the automated update process

  > **Note**
  >
  > The BFB image is available in two formats:
  >
  > - BF-Bundle – includes BlueField firmware, BlueField Arm OS, and DOCA
  > - BF-FW-Bundle – includes BlueField firmware only

- PXE booting the BlueField device with the ISO image provides the same result as BF-Bundle installation using standard installation method for Ubuntu OS

> **Note**

Make sure that the BlueField's firmware version and the DOCA version running on BlueField Arm cores belong to the same release.

Setting BlueField to PXE Boot

Users may set the BlueField UEFI for PXE boot using any of the following methods:

- Direct access to BlueField UEFI menu (see section "Setting Next-boot Device in BlueField UEFI Menu")
- Out-of-band, using the Redfish interface over the BlueField BMC management LAN
- Using `efibootmgr` in the BlueField Linux OS

Regardless of the method, the following is a list of PXE devices in the BlueField UEFI which are compatible with the software upgrade procedure described in this chapter:

- `NET-NIC_P0-IPV4`
- `NET-NIC_P0-IPV6`
- `NET-NIC_P1-IPV4`
- `NET-NIC_P1-IPV6`
- `NET-OOB-IPV4`
- `NET-OOB-IPV6`

Setting Next-boot Device in BlueField UEFI Menu

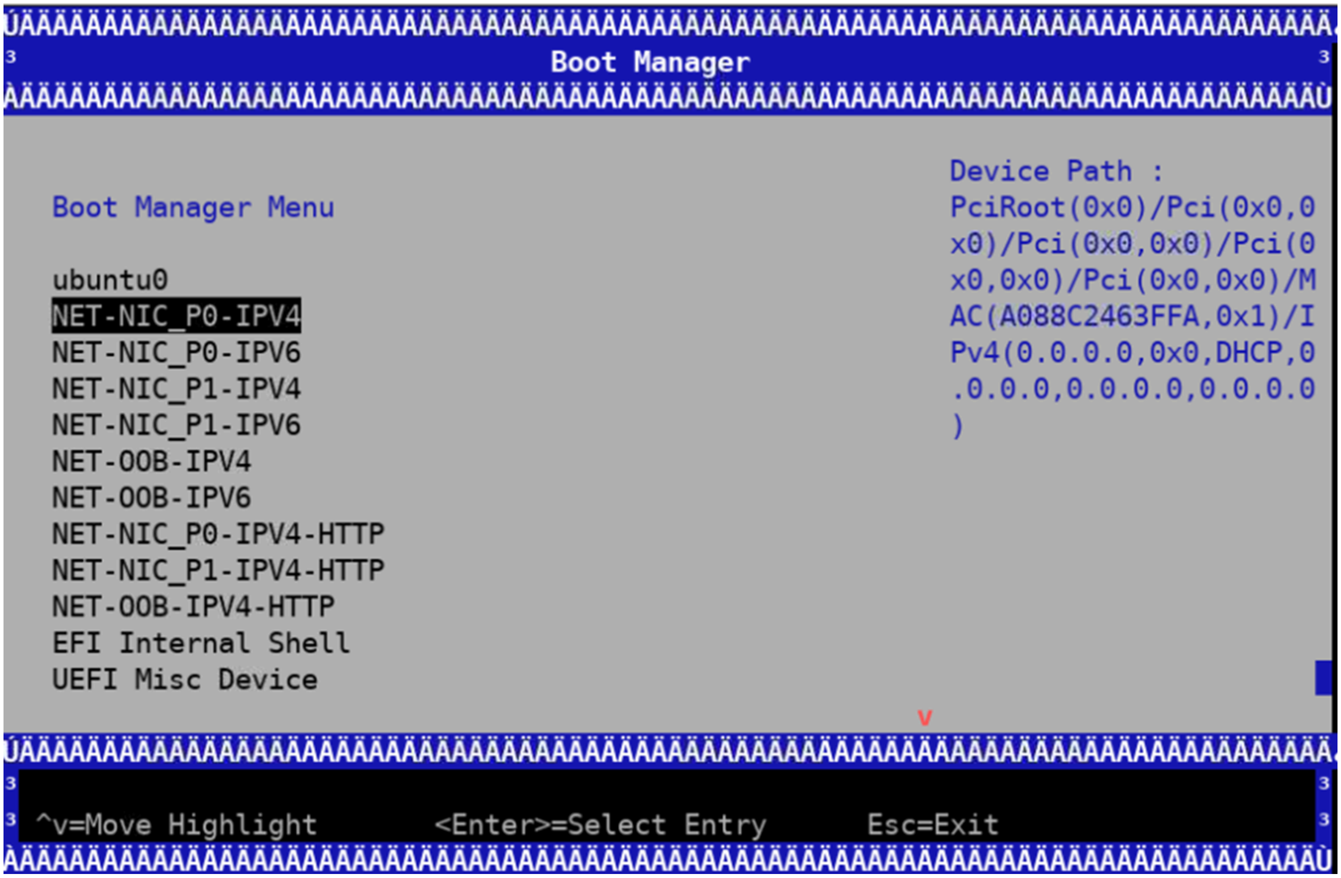
The following steps detail the PXE deployment sequence:

1. Enter into the BlueField UEFI Setup menu.

Info

Refer to section "Accessing the UEFI Menu" for instructions.

2. Navigate to the Boot Manager menu.
3. Select the relevant PXE devices to boot with, depending on how BlueField is connected to the PXE network.



```
                              Boot Manager

   Boot Manager Menu                                  Device Path :
                                                      PciRoot(0x0)/Pci(0x0,0
   ubuntu0                                            x0)/Pci(0x0,0x0)/Pci(0
   NET-NIC_P0-IPV4                                    x0,0x0)/Pci(0x0,0x0)/M
   NET-NIC_P0-IPV6                                    AC(A088C2463FFA,0x1)/I
   NET-NIC_P1-IPV4                                    Pv4(0.0.0.0,0x0,DHCP,0
   NET-NIC_P1-IPV6                                    .0.0.0,0.0.0.0,0.0.0.0
   NET-OOB-IPV4                                       )
   NET-OOB-IPV6
   NET-NIC_P0-IPV4-HTTP
   NET-NIC_P1-IPV4-HTTP
   NET-OOB-IPV4-HTTP
   EFI Internal Shell
   UEFI Misc Device
                                                 v

  ^v=Move Highlight       <Enter>=Select Entry      Esc=Exit
```

Setting BlueField to PXE Boot Using Redfish

To set boot option ID using Redfish:

```
curl -k -u '<username>:<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Settings
-d '{
"Boot": {
        "BootSourceOverrideEnabled" : "Once",
        "BootSourceOverrideMode" : "UEFI",
        "BootSourceOverrideTarget" : "Pxe",
        "UefiTargetBootSourceOverride" : "Boot000",
        "BootNext" : "",
        "AutomaticRetryConfig" : "Disabled"
    }
}' | jq
```

Info

Notice the `UefiTargetBootSourceOverride` and `BootSourceOverrideTarget` fields.

Info

Refer to the *BlueField BMC User Manual* for more information.

Note

To set up a PXE server, please refer to the documentation provided by the distribution vendor. For example, to install Ubuntu 20.04 or later, see official Ubuntu 20.04 documentation.

Setting BlueField to PXE Boot Using efibootmgr

To set boot option ID using efibootmgr:

```
efibootmgr -n 0000
```

This sets a one-time next-boot to device 0000 (`NET-NIC_P0-IPV4`).

PXE/DHCP Server Configuration Tips

BlueField includes information in the DHCP vendor class identifier and vendor specific information options to help the DHCP server identify BlueField and select which boot image to serve (e.g., shim, grub). This can be especially useful for the PXE server to serve a specific shim based on the currently running BlueField OS and shim version to work around the installed UEFI secure boot SBAT entries and to reduce issues with BlueField upgrades and downgrades.

The DHCP vendor class ID will start with `NVIDIA/BF`.

The DHCP vendor specific information will contain:

- BF type (i.e. BF1, BF2, BF3)
- UEFI firmware version
- BF-bundle version

The following is an example of the DHCP vendor information supplied by a BlueField running an older OS version using the Linux `dhcpdump` tool:

```
bf:~# dhcpdump -i tmfifo_net0
...
OPTION:  53 (  1) DHCP message type         1 (DHCPDISCOVER)
...
OPTION:  97 ( 17) UUID/GUID                 009c2debc0368611
..-..6..
```

```
                                                    ee8000a088c20ee8
........
                                                    18                .
OPTION:  94 (  3) Client NDI                       010300           ...
OPTION:  93 (  2) Client System                    000b              ..
OPTION:  60 ( 13) Vendor class identifier        NVIDIA/BF/PXE
OPTION:  43 (131) Vendor specific info            8005424633000081 ..BF3...
                                                    30426c7565466965
0BlueFie
                                                    6c643a342e382e30
ld:4.8.0
                                                    2d322d6765373965 -2-
ge79e
                                                    3037662d64697274 07f-
dirt
                                                    7900000000000000
y.......
                                                    0000000000000000
........
                                                    008248444f43415f
..HDOCA_
                                                    322e352e305f4253
2.5.0_BS
                                                    505f342e352e305f
P_4.5.0_
                                                    5562756e74755f32
Ubuntu_2
                                                    322e30342d312e32 2.04-
1.2
                                                    3032333131303800 0231108.
                                                    0000000000000000
........
                                                    0000000000000000
........
                                                    0000000000000000
........
```

```
                                            000000         ...
...
```

A DHCP server may include some of the following example information in its `dhcpd.conf` to match against the Bluefield above. Note that matching against the vendor specific information using a regex requires setting the substring start offset so that the search occurs after previous strings have ended:

```
...
# Match against vendor class ID
class "PXEClient" {
  match if substring (option vendor-class-identifier, 0, 13) =
"NVIDIA/BF/PXE";
  filename "/shimaa64.efi";
}

# Match against BF Type
class "BfType" {
  match if substring (option vendor-encapsulated-options, 0, 200)
~= "BF3";
  filename "/shimaa64.efi";
}

# Match against UEFI FW version
class "BfFwVer" {
  match if substring (option vendor-encapsulated-options, 6, 200)
~= "4.8.0-2-ge79e07f";
  filename "/shimaa64.efi";
}

# Match against BF Bundle version
class "BfBundleVer" {
  match if substring (option vendor-encapsulated-options, 55, 200)
~= "2.5.0-BSP_4.5.0_Ubuntu_22.04";
```

```
  filename "/shimaa64.efi";
}
...
```

Troubleshooting PXE Boot Issues

More information about how to troubleshoot PXE boot issues can be found in NVIDIA BlueField Troubleshooting Guide.

Deploying BlueField Software Using BFB with PXE

Info

It is recommended to upgrade your BlueField product to the latest software and firmware versions available to benefit from new features and latest bug fixes.

Note

PXE installation is not supported for NIC mode on NVIDIA® BlueField®-3.

The following are the steps to prepare a PXE server to deploy a BFB bundle:

1. Convert the BFB image file to PXE bootable image(s). Run:

   ```
   # mlx-mkbfb -x <BFB>
   ```

For example:

```
# mlx-mkbfb -x DOCA_2.7.0_BSP_4.7.0_Ubuntu_22.04-
<version>.bfb
```

ⓘ **Note**

`mlx-mkbfb` is a Python script that can be found in BlueField release tarball under the `/bin` directory or in the BlueField Arm file system `/usr/bin/mlx-mkbfb`.

2. Copy the output of the 2 files, `dump-image-v0` and `dump-initramfs-v0`, into the PXE server `tftp` path.
3. Create a boot entry in the PXE server. For example:

```
/var/lib/tftpboot/grub.cfg

set default=0
set timeout=5
menuentry 'Bluefield_Ubuntu_22_04_From_BFB' --class red --
class gnu-linux --class gnu --class os {
   linux (tftp)/ubuntu22.04/dump-image-v0 ro ip=dhcp
console=hvc0 console=ttyAMA0
   initrd (tftp)/ubuntu22.04/dump-initramfs-v0
}
```

If additional parameters must be set, use the `bf.cfg` configuration file, then add the `bfks` parameter to the Linux command line in the `grub.cfg` above.

```
menuentry 'Ubuntu22.04 From BFB with bf.cfg' --class red --class gnu-linux --class gnu --class os {
   linux (tftp)/ubuntu22.04/dump-image-v0 console=hvc0 console=ttyAMA0 bfnet=oob_net0:dhcp bfks=http://15.22.82.40/bfks
   initrd (tftp)/ubuntu22.04/dump-initramfs-v0
}
```

`bfks` is a BASH script that runs alongside BFB's `install.sh` script at the beginning of the BFB installation process. Here is an example of `bfks` that creates a `/etc/bf.cfg` file:

```
cat > /etc/bf.cfg << 'EOF'
DEBUG=yes
ubuntu_PASSWORD='$1$3B0RIrfX$TlHry93NFUJzg3Nya00rE1'
EOF
```

4. Define DHCP:

```
/etc/dhcp/dhcpd.conf

allow booting;
allow bootp;

subnet 192.168.100.0 netmask 255.255.255.0 {
  range 192.168.100.10 192.168.100.20;
  option broadcast-address 192.168.100.255;
  option routers 192.168.100.1;
  option domain-name-servers <ip-address-list>
  option domain-search <domain-name-list>;
  next-server 192.168.100.1;
  filename "/BOOTAA64.EFI";
}

# Specify the IP address for this client.
host tmfifo_pxe_client {
  hardware ethernet 00:1a:ca:ff:ff:01;
  fixed-address 192.168.100.2;
}
subnet 20.7.0.0 netmask 255.255.0.0 {
  range 20.7.8.10 20.7.254.254;
  next-server 20.7.6.6;
  filename "/BOOTAA64.EFI";
}
```

Deploying BlueField Software Using ISO with PXE

BlueField software, including Ubuntu OS, NIC firmware and BMC software, can be deployed using an ISO image similar to the standard Ubuntu ISO deployment method. The BlueField ISO image is based on the standard Ubuntu ISO image for Arm64, with an updated kernel and added DOCA packages.

PXE booting the BlueField device with the ISO image results in Arm OS installation (including DOCA packages), NIC firmware and BMC firmware update (if `BMC_PASSWORD` is provided). This is equivalent to using the corresponding version of bf-bundle BFB.

An Ubuntu ISO image can be used if the RShim interface is not available or if there is an existing deployment system in place that can handle a standard Ubuntu ISO image.

PXE Server Setup

Mount the ISO:

```
$ mount bf-bundle-2.7.0085-1-2024-06-14-22-36-50.iso /mnt
$ cp /mnt/casper/vmlinuz /var/lib/tftpboot/boot/
$  cp /mnt/casper/initrd /var/lib/tftpboot/boot/
```

Example of `grub.cfg`:

```
menuentry "Install BF OS" {
    linux /boot/vmlinuz autoinstall fsck.mode=skip no-snapd
console=hvc0 console=ttyAMA0 earlycon=pl011,0x13010000
net.ifnames=0 biosdevname=0 iommu.passthrough=1 ip=dhcp
url=http://<HTTP server IP>/jammy/ISO/bf-bundle-2.7.0085-1-2024-06-14-22-36-50.iso
bfnet=eth0:dhcp bfks=http://<HTTP server IP>/jammy/ISO/bfks
    initrd /boot/initrd
}
```

The `bf.cfg` file can be used to customize the installation procedure. To create `bf.cfg` on the BlueField to be used for the installation use the `bfks` parameter to point to the script located on HTTP server that will create `bf.cfg` file:

bfks example:

```
cat > /etc/bf.cfg << 'EOF'
BMC_PASSWORD="..."
EOF
```

Standard automatic Ubuntu installation using `autoinstall.yaml` is also supported. See Introduction to autoinstall - Ubuntu installation documentation.

Example of `autoinstall.yaml` that can be used to customize the installation and modify `bf.cfg`:

Example of a `grub.cfg` with `autoinstall.yaml`:

```
menuentry "Install BF OS" {
   linux /boot/vmlinuz autoinstall fsck.mode=skip no-snapd console=hvc0 console=ttyAMA0 earlycon=pl011,0x13010000 net.ifnames=0 biosdevname=0 iommu.passthrough=1 ip=dhcp url=http://<HTTP server IP>/jammy/ISO/bf-bundle-2.7.0085-1-2024-06-14-22-36-50.iso force-ai=http://<HTTP server IP>/jammy/ISO/autoinstall.yaml cloud-config-url=/dev/null
    initrd /boot/initrd
}
```

Example of `autoinstall.yaml`:

```
version: 1


apt:
  preserve_sources_list: false
  conf: |
    Dpkg::Options {
      "--force-confdef";
      "--force-confold";
    };

storage:
  swap:
    size: 0
  grub:
    reorder_uefi: true
  config:
  - id: nvme0n1
    type: disk
    ptable: gpt
    path: /dev/nvme0n1
    name: osdisk
    wipe: superblock-recursive
  - id: nvme0n1-part1
    type: partition
    device: nvme0n1
    number: 1
    size: 50MB
    flag: boot
    grub_device: true

  - id: nvme0n1-part1-fs1
    type: format
    fstype: fat32
    label: efi
    volume: nvme0n1-part1

  - id: nvme0n1-part2
    type: partition
    device: nvme0n1
    number: 2
    size: -1

  - id: nvme0n1-part2-fs1
    type: format
    fstype: ext4
    label: root
    volume: nvme0n1-part2

  - id: nvme0n1-mount
    type: mount
    path: /
    device: nvme0n1-part2-fs1
    options: defaults
    passno: 0
    fstype: auto

  - id: nvme0n1-boot-mount
    type: mount
    path: /boot/efi
    device: nvme0n1-part1-fs1
    options: umask=0077
    passno: 1

reporting:
  builtin:
    type: print

# Add user-data so that subiquity doesn't complain about us not
# having a identity section
user-data:
  debug:
    verbose: true
  write_files:
    - path: /etc/iptables/rules.v4
      permissions: '0644'
      owner: 'root:root'
      content: |
        *mangle
        :PREROUTING ACCEPT [45:3582]
        :INPUT ACCEPT [45:3582]
        :FORWARD ACCEPT [0:0]
        :OUTPUT ACCEPT [36:4600]
        :POSTROUTING ACCEPT [36:4600]
        :KUBE-IPTABLES-HINT - [0:0]
        :KUBE-KUBELET-CANARY - [0:0]
        COMMIT
        *filter
        :INPUT ACCEPT [41:3374]
        :FORWARD ACCEPT [0:0]
        :OUTPUT ACCEPT [32:3672]
        :DOCKER-USER - [0:0]
        :KUBE-FIREWALL - [0:0]
        :KUBE-KUBELET-CANARY - [0:0]
        :LOGGING - [0:0]
        :POSTROUTING - [0:0]
        :PREROUTING - [0:0]
        -A INPUT -j KUBE-FIREWALL
        -A INPUT -p tcp -m tcp --dport 111 -j REJECT --reject-with icmp-port-unreachable
        -A INPUT -p udp -m udp --dport 111 -j REJECT --reject-with icmp-port-unreachable
        -A INPUT -i lo -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -d 127.0.0.0/8 -m mark --mark 0xb -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -m mark --mark 0xb -m state --state RELATED,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -p tcp -m tcp ! --dport 22 ! --tcp-flags FIN,SYN,RST,ACK SYN -m mark --mark 0xb -m state --state NEW -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -f -m mark --mark 0xb -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG FIN,SYN,RST,PSH,ACK,URG -m mark --mark 0xb -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG NONE -m mark --mark 0xb -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -m mark --mark 0xb -m state --state INVALID -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -p tcp -m tcp --tcp-flags RST RST -m mark --mark 0xb -m hashlimit --hashlimit-above 2/sec --hashlimit-burst 2 --hashlimit-mode srcip --hashlimit-name hashlimit_0 --hashlimit-htable-expire 30000 -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -p tcp -m mark --mark 0xb -m state --state NEW -m hashlimit --hashlimit-above 50/sec --hashlimit-burst 50 --hashlimit-mode srcip --hashlimit-name hashlimit_1 --hashlimit-htable-expire 30000 -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -p tcp -m mark --mark 0xb -m conntrack --ctstate NEW -m hashlimit --hashlimit-above 60/sec --hashlimit-burst 20 --
```

```
hashlimit-mode srcip --hashlimit-name hashlimit_2 --hashlimit-htable-expire 30000 -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -m mark --mark 0xb -m recent --rcheck --seconds 86400 --name portscan --mask 255.255.255.255 --rsource -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -m mark --mark 0xb -m recent --remove --name portscan --mask 255.255.255.255 --rsource -m comment --comment MD_IPTABLES
        -A INPUT -p tcp -m tcp --dport 22 -m mark --mark 0xb -m conntrack --ctstate NEW -m recent --set --name DEFAULT --mask 255.255.255.255 --rsource -m comment --comment MD_IPTABLES
        -A INPUT -p tcp -m tcp --dport 22 -m mark --mark 0xb -m conntrack --ctstate NEW -m recent --update --seconds 60 --hitcount 50 --name DEFAULT --mask 255.255.255.255 --rsource -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -p tcp -m tcp --dport 443 -m mark --mark 0xb -m conntrack --ctstate NEW -m recent --set --name DEFAULT --mask 255.255.255.255 --rsource -m comment --comment MD_IPTABLES
        -A INPUT -p tcp -m tcp --dport 443 -m mark --mark 0xb -m conntrack --ctstate NEW -m recent --update --seconds 60 --hitcount 10 --name DEFAULT --mask 255.255.255.255 --rsource -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -p udp -m udp --dport 161 -m mark --mark 0xb -m conntrack --ctstate NEW -m recent --set --name DEFAULT --mask 255.255.255.255 --rsource -m comment --comment MD_IPTABLES
        -A INPUT -p udp -m udp --dport 161 -m mark --mark 0xb -m conntrack --ctstate NEW -m recent --update --seconds 60 --hitcount 100 --name DEFAULT --mask 255.255.255.255 --rsource -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -p tcp -m tcp --dport 22 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -p tcp -m tcp --dport 443 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -p tcp -m tcp --dport 179 -m mark --mark 0xb -m
conntrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 68 -m mark --mark 0xb -m
conntrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 122 -m mark --mark 0xb -m
conntrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 161 -m mark --mark 0xb -m
conntrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 6306 -m mark --mark 0xb -m
conntrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 69 -m mark --mark 0xb -m
conntrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 389 -m mark --mark 0xb -m
conntrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
        -A INPUT -p tcp -m tcp --dport 389 -m mark --mark 0xb -m
conntrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 1812:1813 -m mark --mark 0xb
-m conntrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 49 -m mark --mark 0xb -m
conntrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
        -A INPUT -p tcp -m tcp --dport 49 -m mark --mark 0xb -m
conntrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --sport 53 -m mark --mark 0xb -m
conntrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
```

```
        -A INPUT -p tcp -m tcp --sport 53 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 500 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 4500 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 1293 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -p tcp -m tcp --dport 1293 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 1707 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -p tcp -m tcp --dport 1707 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -i lo -p udp -m udp --dport 3786 -m conntrack --ctstate NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -i lo -p udp -m udp --dport 33000 -m conntrack --ctstate NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -p icmp -m mark --mark 0xb -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --sport 5353 --dport 5353 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 33434:33523 -m mark --mark 0xb -m comment --comment MD_IPTABLES -j REJECT --reject-with icmp-port-unreachable
        -A INPUT -p udp -m udp --dport 123 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 514 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 67 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -p tcp -m tcp --dport 60102 -m mark --mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m comment --comment "MD_IPTABLES: Feature HA port" -j ACCEPT
        -A INPUT -m mark --mark 0xb -m comment --comment MD_IPTABLES -j LOGGING
        -A FORWARD -j DOCKER-USER
        -A OUTPUT -o oob_net0 -m comment --comment MD_IPTABLES -j ACCEPT
        -A DOCKER-USER -j RETURN
        -A LOGGING -m mark --mark 0xb -m comment --comment MD_IPTABLES -j NFLOG --nflog-prefix "IPTables-Dropped: " --nflog-group 3
        -A LOGGING -m mark --mark 0xb -m comment --comment MD_IPTABLES -j DROP
        -A PREROUTING -i oob_net0 -m comment --comment MD_IPTABLES -j MARK --set-xmark 0xb/0xffffffff
        -A PREROUTING -p tcp -m tcpmss ! --mss 536:65535 -m tcp ! --dport 22 -m mark --mark 0xb -m conntrack --ctstate NEW -m comment --comment MD_IPTABLES -j DROP
        COMMIT
        *nat
        :PREROUTING ACCEPT [1:320]
        :INPUT ACCEPT [1:320]
        :OUTPUT ACCEPT [8:556]
        :POSTROUTING ACCEPT [8:556]
        :KUBE-KUBELET-CANARY - [0:0]
        :KUBE-MARK-DROP - [0:0]
        :KUBE-MARK-MASQ - [0:0]
        :KUBE-POSTROUTING - [0:0]
        -A POSTROUTING -m comment --comment "kubernetes postrouting rules" -j KUBE-POSTROUTING
        -A KUBE-MARK-DROP -j MARK --set-xmark 0x8000/0x8000
        -A KUBE-MARK-MASQ -j MARK --set-xmark 0x4000/0x4000
        -A KUBE-POSTROUTING -m mark ! --mark 0x4000/0x4000 -j RETURN
        -A KUBE-POSTROUTING -j MARK --set-xmark 0x4000/0x0
        -A KUBE-POSTROUTING -m comment --comment "kubernetes service traffic requiring SNAT" -j MASQUERADE --random-fully
        COMMIT
  users:
    - name: ubuntu
      lock_passwd: False
      groups: adm, audio, cdrom, dialout, dip, floppy, lxd, netdev, plugdev, sudo, video
      sudo: ALL=(ALL) NOPASSWD:ALL
      shell: /bin/bash
      plain_text_passwd: 'ubuntu'
  chpasswd:
    list: |
      ubuntu:ubuntu
    expire: True
  no_ssh_fingerprints: true
  runcmd:
    - [ /usr/sbin/netfilter-persistent, start ]
    - [ /opt/mellanox/doca/services/telemetry/import_doca_telemetry.sh ]
    - [ /usr/bin/bfrshlog, "INFO: DPU is ready" ]

late-commands:
  # write release file
  - |
    cat << EOF > /target/etc/bf-release
    BF_NAME="Mellanox Bluefield"
    BF_PRETTY_NAME="Mellanox Bluefield"
    BF_SWBUILD_TIMESTAMP="2024-06-12-12-47-25"
    BF_SWBUILD_VERSION="2.7.0085-1"
    BF_COMMIT_ID="7fce146"
    BF_PLATFORM="BlueField SoC"
    BF_SERIAL_NUMBER="1332723060006"
    EOF

  # mount cdrom
  - mkdir -p /target/tmp/cdrom
  - mount --bind /cdrom /target/tmp/cdrom || true
  - |
    cat << EOF > /target/etc/apt/sources.list
    deb [check-date=no] file:///tmp/cdrom/ jammy main restricted
    EOF

  # avoid running flash kernel after install kernel
  - mkdir -p /target/run/systemd
  - echo docker > /target/run/systemd/container

  # Install packages
  - curtin in-target -- apt update -y
  - curtin in-target -- apt remove -y --purge `dpkg --list | grep openipmi | awk '{print $2}'`
  - curtin in-target -- /bin/bash -c "DEBIAN_FRONTEND=noninteractive RUN_FW_UPDATER=no apt-get install --no-install-recommends -y acpid bc binutils bridge-utils build-essential cracklib-runtime dc docker.io flash-kernel i2c-tools ifenslave iperf3 iptables-persistent iputils-arping iputils-ping iputils-tracepath kexec-tools libpam-pwquality libssl-dev lldpad lm-sensors net-tools network-manager nfs-common nvme-cli openssh-server python3.10 python3-pyinotify python3-pip rasdaemon rsync sbsigntool shim-signed tcpdump watchdog doca-runtime doca-devel containerd kubelet runc nv-common-apis nvidia-repo-keys linux-bluefield-modules-bluefield linux-image-5.15.0-1042-bluefield"

  # rewrite sources
  - |
    cat << EOF > /target/etc/apt/sources.list
    deb http://ports.ubuntu.com/ubuntu-ports/ jammy main restricted universe multiverse
    deb http://ports.ubuntu.com/ubuntu-ports/ jammy-updates main restricted universe multiverse
    deb http://ports.ubuntu.com/ubuntu-ports/ jammy-security main restricted universe multiverse
    EOF

  # Allow cloud-init to configure networking
  - find /target/etc/cloud/cloud.cfg.d/ -type f ! -name README !
-name 05_logging.cfg ! -name 90_dpkg.cfg -delete || true;
  - curtin in-target -- cloud-init clean

  # Post-installation steps
  # Create bf.cfg
  - |
    cat << EOF > /target/etc/bf.cfg
    # UPDATE_ATF_UEFI - Updated ATF/UEFI (Default: yes)
    # Relevant for PXE installation only as while using RSHIM interface
ATF/UEFI
    # will always be updated using capsule method
    UPDATE_ATF_UEFI="yes"


##################################################################
    # BMC Component Update

##################################################################
    # BMC_USER - User name to be used to access BMC (Default:
root)
    BMC_USER="root"

    # BMC_PASSWORD - Password used by the BMC user to access BMC
(Default: None)
    BMC_PASSWORD=""

    # BMC_IP_TIMEOUT - Maximum time in seconds to wait for the
connection to the
    # BMC to be established (Default: 600)
    BMC_IP_TIMEOUT=600
    # BMC_TASK_TIMEOUT - Maximum time in seconds to wait for BMC task (BMC/CEC
    # Firmware update) to complete (Default: 1800)
    BMC_TASK_TIMEOUT=1800

    # UPDATE_BMC_FW - Update BMC firmware (Default: yes)
    UPDATE_BMC_FW="yes"

    # BMC_REBOOT - Reboot BMC after BMC firmware update to apply the new version
    # (Default: no). Note that the BMC reboot will reset the BMC console.
    BMC_REBOOT="no"

    # UPDATE_CEC_FW - Update CEC firmware (Default: yes)
    UPDATE_CEC_FW="yes"

    # UPDATE_DPU_GOLDEN_IMAGE - Update BlueField Golden Image (Default: yes)
    UPDATE_DPU_GOLDEN_IMAGE="yes"

    # UPDATE_NIC_FW_GOLDEN_IMAGE- Update NIC firmware Golden Image (Default: yes)
    UPDATE_NIC_FW_GOLDEN_IMAGE="yes"

    # pre_bmc_components_update - Shell function called by BFB's install.sh before
    # updating BMC components (no communication to the BMC is established at this
    # point)

    # post_bmc_components_update - Shell function called by BFB's install.sh after
    # updating BMC components
```

```
##################################################################
    # NIC Firmware update

##################################################################
    # WITH_NIC_FW_UPDATE - Update NIC Firmware (Default: no)
    WITH_NIC_FW_UPDATE="yes"
    EOF

    # Run post-installation script to update ATF/UEFI, NIC
firmware and BMC components
  - curtin in-target -- /bin/bash -c "device=/dev/nvme0n1 /usr/local/sbin/bfiso-
post-install.sh || true"

  - curtin in-target -- systemctl disable snapd
```

PXE Sequence with Redfish

HTTP boot configuration can be done using the BlueField BMC's Redfish interface.

ISO upgrade via Redfish to set UEFI HTTPs/PXE boot by setting UEFI first boot source.

To set the UEFI first boot source using Redfish:

1. Follow the instructions under section "Deploying BlueField Software Using BFB with PXE".
2. Check the current boot override settings by performing a GET on the `ComputerSystem` schema over 1GbE to the BlueField BMC. Look for the `"Boot"` property.

```
curl -k -X GET -u root:<password> https://<BF-BMC-
IP>/redfish/v1/Systems/<SystemID>/ | python3 -m json.tool
{
...
"Boot": {
        "BootNext": "",
        "BootOrderPropertySelection": "BootOrder",
        "BootSourceOverrideEnabled": "Disabled",
        "BootSourceOverrideMode": "UEFI",
        "BootSourceOverrideTarget": "None",
        "UefiTargetBootSourceOverride": "None",
        .....
        },
  ....
  "BootSourceOverrideEnabled@Redfish.AllowableValues": [
            "Once",
            "Continuous",
            "Disabled"
        ],
    "BootSourceOverrideTarget@Redfish.AllowableValues": [
            "None",
            "Pxe",
            "UefiHttp",
            "UefiShell",
            "UefiTarget",
            "UefiBootNext"
        ],
  ....
}
```

Info

Boot override enables overriding the first boot source, either once or continuously.

3. The example output above shows the `BootSourceOverrideEnabled` property is `Disabled` and `BootSourceOverrideTarget` is `None`. The `BootSourceOverrideMode` property should always be set to `UEFI`. Allowable

values of `BootSourceOverrideEnabled` and `BootSourceOverrideTarget` are defined in the metadata (`BootSourceOverrideEnabled@Redfish.AllowableValues` and `BootSourceOverrideTarget@Redfish.AllowableValues` respectively).

4. If `BootSourceOverrideEnabled` is set to `Once`, then boot override is disabled after the first boot, and any related properties are reset to their former values to avoid repetition. If it is set to `Continuous`**,** then on every reboot the BlueField keeps performing boot override (HTTPBoot).

5. To perform boot override, perform a PATCH to pending settings URI over the 1GbE to the BlueField BMC.

```
curl -k -X PATCH -d '{"Boot":
{"BootSourceOverrideEnabled":"Once",
"BootSourceOverrideMode":"UEFI", "BootSourceOverrideTarget":
"UefiHttp", "HttpBootUri":"http://<HTTP-Server-
Ip>/Image.iso"}}' -u root:<password> https://<BF-BMC-
IP>/redfish/v1/Systems/<SystemID>/Settings | python3 -m
json.tool
```

For example:

```
curl -k -X GET -u root:<password> https://<BF-BMC-
IP>/redfish/v1/Systems/<SystemID>/ | python3 -m json.tool
{
...
"Boot": {
        "BootNext": "",
        "BootOrderPropertySelection": "BootOrder",
        "BootSourceOverrideEnabled": "Once",
        "BootSourceOverrideMode": "UEFI",
        "BootSourceOverrideTarget": "UefiHttp",
        "UefiTargetBootSourceOverride": "None",
        .....
        },
    .....
}
```

6. After performing the above PATCH successfully, reboot the BlueField using the Redfish Manager schema over the 1GbE to the BlueField BMC:

```
curl -k -u root:<password> -H "Content-Type:
application/json" -X POST https://<BF-BMC-
IP>/redfish/v1/Systems/Bluefield/Actions/ComputerSystem.Reset
-d '{"ResetType" : "GracefulRestart"}'
```

7. Once UEFI has completed, check whether the settings are applied by performing a GET on `ComputerSystem` schema over the 1GbE OOB to the BlueField BMC.

Note

The `HttpBootUri` property is parsed by the Redfish server and the URI is presented to the BlueField as part of DHCP lease when the BlueField performs the HTTP boot.

Customizing BlueField Software Deployment

`bf.cfg` is an optional configuration file which may be used to customize the software deployment process on NVIDIA® BlueField® networking platforms (DPU or SuperNIC).

Note

To update the BMC components, it is required to provide the `BMC_PASSWORD` using `bf.cfg` to the BFB/ISO installation environment.

There are different ways to pass `bf.cfg` along with the BFB or ISO to customize the installation procedure:

- With BFB from the host:

  ```
  # bfb-install -r <rshim device> -c <path to bf.cfg> -b <BFB>
  ```

- Using cat command:

  ```
  # cat <BFB> <path to bf.cfg> > /dev/<rshim device>/boot
  ```

- By appending `bf.cfg` to the BFB and push it to RShim device on a host or BMC:

```
# cat <BFB> <path to bf.cfg> > <new BFB>
```

- In PXE environment using `bfks` parameter to provide a script that will be downloaded by the installation process and run on the Bluefield side at the beginning of installation:

```
cat > /etc/bf.cfg << 'EOF'
BMC_PASSWORD="…"
EOF
```

- Or using `autoinstall.yaml`. See " Deploying BlueField Software Using ISO with PXE" for details.

Changing Default Credentials for "ubuntu" User via bf.cfg

Info

For a comprehensive list of the supported parameters to customize `bf.cfg` during BFB installation, refer to section "bf.cfg Parameters".

Ubuntu users are prompted to change the default password (ubuntu) for the default user (ubuntu) upon first login. Logging in will not be possible even if the login prompt appears until all services are up ("`DPU is ready`" message appears in `/dev/rshim0/misc`).

Note

Attempting to log in before all services are up prints the following message: `Permission denied, please try again.`

Alternatively, Ubuntu users can provide a unique password that will be applied at the end of the BFB installation. This password must be defined in a `bf.cfg` configuration file. To set the password for the `ubuntu` user:

1. Create password hash. Run:

   ```
   # openssl passwd -1
   Password:
   Verifying - Password:
   $1$3B0RIrfX$TlHry93NFUJzg3Nya00rE1
   ```

2. Add the password hash in quotes to the `bf.cfg` file:

   ```
   # vim bf.cfg
   ubuntu_PASSWORD='$1$3B0RIrfX$TlHry93NFUJzg3Nya00rE1'
   ```

   The `bf.cfg` file is used with the `bfb-install` script in the steps that follow.

Changing UEFI Password Using bf.cfg

To change the UEFI password, add the current UEFI password under parameter UEFI_PASSWORD and define the new UEFI password under NEW_UEFI_PASSWORD inside the `bf.cfg` configuration file.

Changing BMC Password Using bf.cfg

To change the BMC root password, add the current BMC root password under parameter BMC_PASSWORD and define the new BMC root password under NEW_BMC_PASSWORD inside the `bf.cfg` configuration file.

Advanced Customizations During BFB Installation

Using special purpose configuration parameters in the `bf.cfg` file, the BlueField's boot options and OS can be further customized. For a full list of the supported parameters to customize your BlueField during BFB installation, refer to section "bf.cfg Parameters". In addition, the `bf.cfg` file offers further control on customization of BlueField OS installation and software configuration through scripting.

Add any of the following functions to the `bf.cfg` file for them to be called by the `install.sh` script embedded in the BFB:

- `bfb_modify_os` – called after the file system is extracted on the target partitions. It can be used to modify files or create new files on the target file system mounted under `/mnt`. So the file path should look as follows: `/mnt/<expected_path_on_target_OS>`. This can be used to run a specific tool from the target OS (remember to add `/mnt` to the path for the tool).
- `bfb_pre_install` – called before eMMC/SSD partitions format and OS filesystem is extracted
- `bfb_post_install` – called as a last step before reboot. All eMMC/SSD partitions are unmounted at this stage.

For example, the `bf.cfg` script below disables OVS bridge creation upon boot:

```
# cat /root/bf.cfg

bfb_modify_os()
{
        log ===================== bfb_modify_os
=====================
        log "Disable OVS bridges creation upon boot"
        sed -i -r -e 's/(CREATE_OVS_BRIDGES=).*/\1"no"/'
/mnt/etc/mellanox/mlnx-ovs.conf
}

bfb_pre_install()
{
        log ===================== bfb_pre_install
=====================
}
```

```
bfb_post_install()
{
        log ===================== bfb_post_install
=====================
}
```

bf.cfg Parameters

The following is a comprehensive list of the supported parameters to customize the `bf.cfg` file for BFB installation:

```

##################################################################
# Configuration which can also be set in
#   UEFI->Device Manager->System Configuration
##################################################################
# Enable SMMU in ACPI.
#SYS_ENABLE_SMMU = TRUE

# Enable I2C0 in ACPI.
#SYS_ENABLE_I2C0 = FALSE

# Disable SPMI in ACPI.
#SYS_DISABLE_SPMI = FALSE

# Enable the second eMMC card which is only available on the
BlueField Reference Platform.
#SYS_ENABLE_2ND_EMMC = FALSE

# Enable eMMC boot partition protection.
#SYS_BOOT_PROTECT = FALSE

# Enable SPCR table in ACPI.
#SYS_ENABLE_SPCR = FALSE
```

```
# Disable PCIe in ACPI.
#SYS_DISABLE_PCIE = FALSE

# Enable OP-TEE in ACPI.
#SYS_ENABLE_OPTEE = FALSE

##################################################################
# Boot Order configuration
# Each entry BOOT<N> could have the following format:
# PXE:
#   BOOT<N> = NET-<NIC_P0 | NIC_P1 | OOB | RSHIM>-<IPV4 | IPV6>
# PXE over VLAN (vlan-id in decimal):
#   BOOT<N> = NET-<NIC_P0 | NIC_P1 | OOB | RSHIM>[.<vlan-id>]-<IPV4 | IPV6>
# UEFI Shell:
#   BOOT<N> = UEFI_SHELL
# DISK: boot entries created during OS installation.
#   BOOT<N> = DISK
##################################################################
# This example configures PXE boot over the 2nd ConnectX port.
# If fails, it continues to boot from disk with boot entries created during OS
# installation.
#BOOT0 = NET-NIC_P1-IPV4
#BOOT1 = DISK

# UPDATE_ATF_UEFI - Updated ATF/UEFI (Default: yes)
# Relevant for PXE installation only as while using RSHIM interface ATF/UEFI
# will always be updated using capsule method
UPDATE_ATF_UEFI="yes"

# To change UEFI password set UEFI_PASSWORD to its current value and NEW_UEFI_PASSWORD to the new UEFI password (clear text).
UEFI_PASSWORD=<current UEFI password>
NEW_UEFI_PASSWORD=<new UEFI password>
```

```
# UPDATE_DPU_OS - Update/Install BlueField Operating System
(Default: yes)
UPDATE_DPU_OS="yes"

# grub_admin_PASSWORD - Hashed password to be set for the "admin"
user to enter Grub menu
# Relevant for Ubuntu BFB only. (Default: is not set)
# E.g.:
grub_admin_PASSWORD='grub.pbkdf2.sha512.10000.5EB1FF92FDD89BDAF339
grub_admin_PASSWORD='grub.pbkdf2.sha512.10000.<hashed password>'

# ubuntu_PASSWORD - Hashed password to be set for "ubuntu" user
during BFB installation process.
# Relevant for Ubuntu BFB only. (Default: is not set)
ubuntu_PASSWORD=<hashed password>

##################################################################
# BMC Component Update
##################################################################
# BMC_USER - User name to be used to access BMC (Default: root)
BMC_USER="root"

# BMC_PASSWORD - Password used by the BMC user to access BMC
(Default: None)
BMC_PASSWORD=""

# NEW_BMC_PASSWORD - can be used to change BMC_PASSWORD to the
new one (Default: None)
# Note: current BMC_PASSWORD is required
NEW_BMC_PASSWORD=<new BMC password>

# BMC_SSH_USER - User name to be used to access BMC using ssh
(Default: same as BMC_USER)
BMC_SSH_USER="root"
```

```
# BMC_SSH_PASSWORD - Password used by the BMC user to access BMC
using ssh (Default: same as BMC_PASSWORD)
BMC_SSH_PASSWORD=""   # BMC_IP_TIMEOUT - Maximum time in seconds
to wait for the connection to the

# BMC to be established (Default: 600)
BMC_IP_TIMEOUT=600

# BMC_TASK_TIMEOUT - Maximum time in seconds to wait for BMC task
(BMC/CEC
# Firmware update) to complete (Default: 1800)
BMC_TASK_TIMEOUT=1800

# UPDATE_BMC_FW - Update BMC firmware (Default: yes)
UPDATE_BMC_FW="yes"

# BMC_REBOOT - Reboot BMC after BMC firmware update to apply the
new version
# (Default: no). Note that the BMC reboot will reset the BMC
console.
BMC_REBOOT="no"

# UPDATE_CEC_FW - Update CEC firmware (Default: yes)
UPDATE_CEC_FW="yes"

# CEC_REBOOT - Reboot CEC after CEC firmware update to apply the
new version (Default: no).
# Note: CEC_REBOOT is supported only if currently installed CEC
firmware version is 00.02.0180.0000 or newer.
# Otherwise, Host power cycle will be required to apply the new
CEC firmware.
CEC_REBOOT="no"

# UPDATE_DPU_GOLDEN_IMAGE - Update BlueField Golden Image
(Default: yes)
UPDATE_DPU_GOLDEN_IMAGE="yes"
```

```
# UPDATE_NIC_FW_GOLDEN_IMAGE- Update NIC firmware Golden Image (Default: yes)
UPDATE_NIC_FW_GOLDEN_IMAGE="yes"

# pre_bmc_components_update - Shell function called by BFB's install.sh before
# updating BMC components (no communication to the BMC is established at this
# point)

# post_bmc_components_update - Shell function called by BFB's install.sh after
# updating BMC components

##################################################################
# NIC Firmware update
##################################################################
# WITH_NIC_FW_UPDATE - Update NIC Firmware (Default: yes)
WITH_NIC_FW_UPDATE="yes"

##################################################################
# Other misc configuration
##################################################################

# MAC address of the rshim network interface (tmfifo_net0).
#NET_RSHIM_MAC = 00:1a:ca:ff:ff:01

# DHCP class identifier for PXE (arbitrary string up to 32 characters)
#PXE_DHCP_CLASS_ID = NVIDIA/BF/PXE

# Create dual boot partition scheme (Ubuntu only)
# DUAL_BOOT=yes

# Upgrade NIC firmware
# WITH_NIC_FW_UPDATE=yes

# Target storage device for the BlueField Arm OS (Default SSD:
/dev/nvme0n1)
device=/dev/nvme0n1

# bfb_modify_os - SHELL function called after the file system is
extracted on the target partitions.
# It can be used to modify files or create new files on the
target file system mounted under
# /mnt. So the file path should look as follows:
/mnt/<expected_path_on_target_OS>. This
# can be used to run a specific tool from the target OS (remember
to add /mnt to the path for
# the tool).

# bfb_pre_install - SHELL function called before partitions
format
# and OS filesystem is extracted

# bfb_post_install - SHELL function called as a last step before
reboot.
# All partitions are unmounted at this stage.
```

System Configuration Dump

The `bfcfg` script included with the BlueField Arm OS can be used to dump system configuration information modified with `bf.cfg` or through the UEFI menu. The `-d` parameter can be used to enable dump mode and the `-l` parameter used to set the dump level (how much configuration information is logged).

The following is an example of a full system configuration dump:

```
root@bu-lab102s2-oob:~# bfcfg -d -l 2
```

```
icm: LARGE_ICM_SIZE=0x200
mfg: MFG_OOB_MAC=a0:88:c2:0e:88:12
mfg: MFG_OPN=900-9D3B4-00EN-EAA
mfg: MFG_SKU=900-9D3B4-00EN-EAA
mfg: MFG_MODL=D3B4
mfg: MFG_SN=MT2329XZ0117
mfg: MFG_UUID=0a91f4868e2eee118000a088c20e87fe
mfg: MFG_REV=A9
sys: ENABLE_SMMU=1
sys: DISABLE_SPMI=0
sys: ENABLE_2ND_EMMC=0
sys: BOOT_PROTECT=0
sys: ENABLE_SPCR=0
sys: DISABLE_PCIE=0
sys: ENABLE_OPTEE=0
sys: DISABLE_TMFF=0
sys: ENABLE_I2C0=0
sys: DISABLE_FORCE_PXE_RETRY=0
sys: ENABLE_BMC_FIELD_MODE=0
sys: LARGE_ICMC_SIZE=0x200
sys: CE_THRESHOLD=0x1388
sys: DISABLE_HEST=0
sys: L3_CACHE_PART_LEVEL=0x0
sys: ENABLE_I2C3=0
sys: ENABLE_FORCE_BOOT_RETRY=0
sys: ENABLE_OEM_MFG_CONFIG=0
sys: DISABLE_I2C1=0
sys: DISABLE_AUTO_BOOT_REFRESH=0
sys: DISPLAY_BMC_NET_CONFIG=0
sys: SKIP_REDFISH=0
sys: ENABLE_REDFISH=1
sys: RTCSYNC=0
misc: NET_RSHIM_MAC=00:1a:ca:ff:ff:01
misc:
PXE_DHCP_CLASS_ID=00:00:16:47:4e:56:49:44:49:41:2f:42:46:2f:50:58:45
misc: BF_BUNDLE_VERSION=
boot: BOOT_TIMEOUT_SEC=3
boot: BOOT_CURRENT=0006
boot:
  BOOT_ORDER:
    0006,0000,0001,0002,0003,0004,0005,0007,
    0008,0009,000A,000B,000C,000D,000E,000F,
    0010,0011,0012,0013,0014,0015,0016,0017,
    0018,0019,001A,001B
  BOOT_OPTIONS:
    Boot0000* NET-NIC_P0-IPV4
    Boot0001* NET-NIC_P0-IPV6
    Boot0002* NET-OOB-IPV4
    Boot0003* NET-OOB-IPV6
    Boot0004* NET-NIC_P0-IPV4-HTTP
    Boot0005* NET-OOB-IPV4-HTTP
    Boot0006* ubuntu0
    Boot0007* UiApp
    Boot0008* EFI Internal Shell
    Boot0009* UEFI Misc Device
    Boot000A* UEFI MTFDHBL128TDP 22303A1B7B73 1
    Boot000B* UEFI Non-Block Boot Device
    Boot000C* UEFI PXEv4 (MAC:001ACAFFFF01)
    Boot000D* UEFI PXEv6 (MAC:001ACAFFFF01)
    Boot000E* UEFI HTTPv4 (MAC:001ACAFFFF01)
    Boot000F* UEFI HTTPv6 (MAC:001ACAFFFF01)
    Boot0010* UEFI PXEv4 (MAC:A088C20E8812 VLAN4040)
    Boot0011* UEFI PXEv6 (MAC:A088C20E8812 VLAN4040)
    Boot0012* UEFI HTTPv4 (MAC:A088C20E8812 VLAN4040)
    Boot0013* UEFI HTTPv6 (MAC:A088C20E8812 VLAN4040)
    Boot0014* UEFI PXEv4 (MAC:A088C20E8806)
    Boot0015* UEFI PXEv6 (MAC:A088C20E8806)
    Boot0016* UEFI HTTPv4 (MAC:A088C20E8806)
    Boot0017* UEFI HTTPv6 (MAC:A088C20E8806)
    Boot0018* UEFI PXEv4 (MAC:A088C20E8812)
    Boot0019* UEFI PXEv6 (MAC:A088C20E8812)
    Boot001A* UEFI HTTPv4 (MAC:A088C20E8812)
    Boot001B* UEFI HTTPv6 (MAC:A088C20E8812)
sb: SECURE_BOOT_ENABLED=0
sb: SETUP_MODE=SETUP
sb:
  Platform Key (PK):

  Key Exchange Key (KEK):

  Signatures Database (DB):
    [key 1]
    SHA1 Fingerprint:
ed:11:d7:44:35:92:8a:74:aa:34:d5:1a:dc:73:c6:a1:a8:a9:ec:23
    Certificate:
        Data:
            Version: 3 (0x2)
            Serial Number:
                ee:e2:30:01:50:fe:10:82
            Signature Algorithm: sha256WithRSAEncryption
            Issuer: CN=Bluefield Signature Database Test Key
            Validity
                Not Before: Nov 12 16:49:55 2020 GMT
                Not After : Nov 10 16:49:55 2030 GMT
            Subject: CN=Bluefield Signature Database Test Key
            Subject Public Key Info:
                Public Key Algorithm: rsaEncryption
                    Public-Key: (2048 bit)
                    Modulus:

00:ba:6d:83:59:1a:9c:9b:64:c2:a4:1e:fc:1a:a6:

cc:44:6a:61:e2:fa:35:e1:53:ca:ee:3c:7d:7b:a4:

21:8b:46:b3:2a:98:91:4a:ce:6a:27:9e:5c:47:40:

e3:b3:94:fc:1d:9f:2f:26:0a:04:ac:65:c1:07:12:
                        4c:0d:63:d6:bb:85:12:02:f8:77:30:b3:35:06:71:
```

```
91:1e:80:0c:19:96:5a:6d:41:40:9e:13:7d:5f:fe:

d9:40:b5:2d:ed:e9:9e:e5:27:aa:e4:69:43:24:59:

cf:26:2a:fe:d7:04:04:de:fe:d6:47:20:18:e6:99:

aa:f7:44:14:9c:e3:61:b3:6d:2a:38:b1:f3:dc:19:

0e:59:32:e0:9a:ca:6e:6a:67:b4:a7:18:d5:1a:50:

65:e7:29:35:4b:63:55:b2:97:52:cc:0d:e8:76:d2:

fc:7a:f8:58:4c:c1:59:00:36:6e:b8:e9:03:34:1f:

be:07:9f:6b:26:65:c5:1c:c0:05:d3:11:d6:cc:11:

e8:2f:b5:73:c8:8f:19:b8:b8:a5:d4:19:3a:e3:90:

7c:61:b1:77:ac:e7:f8:83:22:bd:9f:46:d3:fe:2a:

f6:3d:92:83:1e:95:3d:80:1e:1c:47:70:80:ca:3e:

d2:8f:5a:1c:0c:e9:0e:91:25:85:24:98:78:66:a1:
                        5f:21
                    Exponent: 65537 (0x10001)
            X509v3 extensions:
                X509v3 Subject Key Identifier:

A1:09:F0:17:07:BA:67:69:C4:D5:46:53:0B:A1:59:2C:7D:AE:DC:3B
                X509v3 Authority Key Identifier:

A1:09:F0:17:07:BA:67:69:C4:D5:46:53:0B:A1:59:2C:7D:AE:DC:3B
                X509v3 Basic Constraints: critical
                    CA:TRUE
        Signature Algorithm: sha256WithRSAEncryption
        Signature Value:
            58:3d:15:d1:ae:33:46:5c:6f:99:9c:55:a3:06:cb:fd:36:68:
            06:ab:1b:e5:79:8f:7b:47:86:0c:c7:04:ea:b5:1c:a4:f5:96:
            88:dc:98:af:8f:db:bb:4f:8b:27:4b:ef:a4:11:4d:3d:8a:ea:
            ad:d9:13:85:85:82:9d:23:9c:6b:d0:35:f1:f1:a5:dd:7c:6c:
            d1:84:4e:a1:b0:ac:7e:f6:db:cb:9f:8b:b7:ce:a5:12:78:07:
            ed:ea:95:e4:2b:b7:60:93:c9:5a:80:21:b7:53:8b:7e:7d:ea:
            29:31:14:73:01:e2:88:ea:20:2f:56:b7:e3:2b:85:15:09:b2:
            7f:19:86:cd:e0:c4:71:ff:2c:80:d3:b6:80:ec:97:7c:0e:47:
            93:fe:ad:df:c6:87:67:8b:62:74:38:60:4f:a7:90:05:1e:bb:
            ac:6e:51:23:74:c5:c9:90:ac:12:7f:d9:a3:d6:56:87:23:d7:
            88:2d:9e:1f:8b:8e:45:16:95:7d:ef:6f:43:00:95:2d:47:26:
            43:6c:9e:fc:ec:3b:04:3d:2c:18:c4:4c:2a:15:3a:18:18:f4:
            49:11:77:74:29:cc:19:c6:45:15:47:0c:2a:99:4c:22:51:34:
            24:12:94:fd:df:38:49:55:e1:44:b9:7c:f5:49:a2:70:26:64:
            ec:01:c6:64

    [key 2]
    SHA1 Fingerprint:
79:a1:d2:9e:8a:cf:31:f0:91:bc:51:b1:bc:d0:47:b1:7d:69:cd:9e
    Certificate:
        Data:
            Version: 3 (0x2)
            Serial Number:
                aa:8d:31:99:63:ff:50:c0
            Signature Algorithm: sha256WithRSAEncryption
            Issuer: O=Mellanox Technologies, CN=Mellanox
Technologies signing key/emailAddress=support@mellanox.com
            Validity
                Not Before: Sep  4 16:12:47 2020 GMT
                Not After : Aug 11 16:12:47 2120 GMT
            Subject: O=Mellanox Technologies, CN=Mellanox
Technologies signing key/emailAddress=support@mellanox.com
            Subject Public Key Info:
                Public Key Algorithm: rsaEncryption
                    Public-Key: (2048 bit)
```

```
             Modulus:

00:e1:1b:fa:53:4c:4b:ef:2a:d3:8e:93:d8:99:e0:

a1:81:ff:6a:9c:87:2f:b0:d9:8b:b8:40:8c:3f:00:

3b:e1:b1:eb:6c:bf:30:cb:83:25:ce:fd:f6:a6:b9:

01:f7:44:ab:e5:a4:65:11:a6:36:6b:c1:23:08:2c:

65:50:25:38:2a:4b:8c:91:fd:02:cf:4c:e4:86:93:

b0:50:e7:7d:65:a0:ab:51:50:54:bb:e6:3d:85:db:

94:91:93:83:0a:f1:70:aa:4d:ad:17:41:d4:e7:ab:

f8:65:23:da:3c:0c:eb:9d:09:26:8e:42:8e:3f:e5:

50:ec:4d:d5:2e:38:09:39:6b:26:2e:b2:68:4b:fe:

fa:1a:eb:79:e0:5d:da:52:bf:d7:c4:d2:fd:3f:0d:

63:7a:e6:7e:fd:e6:41:b2:d6:b6:d8:c1:17:11:eb:

a5:bf:04:f3:9f:c6:bf:18:7f:3e:8d:44:93:26:73:

f4:e1:33:82:3a:81:3f:54:8b:7c:83:0e:ca:b7:9c:

02:97:8b:cf:aa:1f:b8:d0:bb:39:27:cb:50:b0:fa:
                ad:f0:e2:e1:69:c1:70:99:01:42:72:20:03:91:1c:

2b:78:a1:db:83:35:d5:ae:be:fc:42:c7:7a:db:59:

bd:f0:6d:42:c8:39:b1:5b:c3:e4:3a:c8:c2:56:64:
                12:39
             Exponent: 65537 (0x10001)
```

```
            X509v3 extensions:
                X509v3 Basic Constraints: critical
                    CA:FALSE
                X509v3 Key Usage:
                    Digital Signature
                X509v3 Subject Key Identifier:

20:12:E5:12:26:69:FF:C0:CC:28:82:7C:61:34:32:9A:6B:EC:0B:88
                X509v3 Authority Key Identifier:

20:12:E5:12:26:69:FF:C0:CC:28:82:7C:61:34:32:9A:6B:EC:0B:88
        Signature Algorithm: sha256WithRSAEncryption
        Signature Value:
            c4:da:d8:15:a4:37:f0:93:84:97:de:b8:0f:7b:8d:8b:ec:dd:
            bb:5f:8d:be:e5:67:a0:22:b6:5a:07:95:15:5b:cd:4e:ab:a9:
            1f:2e:c8:f9:49:45:bc:f2:97:ad:6e:bd:16:8c:e1:1a:be:c2:
            8a:9c:39:59:93:76:36:08:24:15:ff:0c:a4:3c:50:d9:75:b9:
            06:03:97:17:68:07:0e:c6:d0:0d:3c:7e:c4:76:d6:de:20:80:
            36:ea:48:11:41:93:3c:d3:b8:c8:dc:e3:d8:28:d9:de:af:ba:
            14:aa:98:a9:05:19:c0:d7:a9:db:07:63:f6:07:1f:f0:e0:b8:
            ef:49:65:6d:d9:a6:1f:83:9d:2d:b2:49:74:63:fa:2b:f8:a4:
            02:02:eb:ab:b2:2f:91:1a:91:f4:58:3f:85:5b:aa:92:11:37:
            31:d9:2a:81:43:d8:1d:77:89:a5:54:2b:be:dd:58:91:57:b1:
            a5:2b:75:cc:78:ce:7e:5e:b9:22:ca:fb:c3:bb:d6:2d:7c:90:
            ed:e0:7b:e8:bb:8a:e6:f2:44:60:58:74:e9:a5:de:19:a4:a6:
            d1:43:e2:a3:ac:97:b4:b7:94:4f:93:d7:54:2d:07:c2:d0:c1:
            94:47:0c:7f:f3:0a:5f:f6:06:8f:4d:1d:ec:01:b0:9b:e1:35:
            e3:8c:e4:f9

    [key 3]
    SHA1 Fingerprint:
46:de:f6:3b:5c:e6:1c:f8:ba:0d:e2:e6:63:9c:10:19:d0:ed:14:f3
    Certificate:
        Data:
            Version: 3 (0x2)
            Serial Number:
                61:08:d3:c4:00:00:00:00:00:04
            Signature Algorithm: sha256WithRSAEncryption
            Issuer: C=US, ST=Washington, L=Redmond, O=Microsoft
Corporation, CN=Microsoft Corporation Third Party Marketplace
Root
            Validity
                Not Before: Jun 27 21:22:45 2011 GMT
                Not After : Jun 27 21:32:45 2026 GMT
            Subject: C=US, ST=Washington, L=Redmond, O=Microsoft
Corporation, CN=Microsoft Corporation UEFI CA 2011
            Subject Public Key Info:
                Public Key Algorithm: rsaEncryption
                    Public-Key: (2048 bit)
                    Modulus:

00:a5:08:6c:4c:c7:45:09:6a:4b:0c:a4:c0:87:7f:
                        06:75:0c:43:01:54:64:e0:16:7f:07:ed:92:7d:0b:

b2:73:bf:0c:0a:c6:4a:45:61:a0:c5:16:2d:96:d3:

f5:2b:a0:fb:4d:49:9b:41:80:90:3c:b9:54:fd:e6:

bc:d1:9d:c4:a4:18:8a:7f:41:8a:5c:59:83:68:32:

bb:8c:47:c9:ee:71:bc:21:4f:9a:8a:7c:ff:44:3f:

8d:8f:32:b2:26:48:ae:75:b5:ee:c9:4c:1e:4a:19:

7e:e4:82:9a:1d:78:77:4d:0c:b0:bd:f6:0f:d3:16:

d3:bc:fa:2b:a5:51:38:5d:f5:fb:ba:db:78:02:db:

ff:ec:0a:1b:96:d5:83:b8:19:13:e9:b6:c0:7b:40:

7b:e1:1f:28:27:c9:fa:ef:56:5e:1c:e6:7e:94:7e:
```

```
c0:f0:44:b2:79:39:e5:da:b2:62:8b:4d:bf:38:70:

e2:68:24:14:c9:33:a4:08:37:d5:58:69:5e:d3:7c:

ed:c1:04:53:08:e7:4e:b0:2a:87:63:08:61:6f:63:

15:59:ea:b2:2b:79:d7:0c:61:67:8a:5b:fd:5e:ad:
                       87:7f:ba:86:67:4f:71:58:12:22:04:22:22:ce:8b:
                       ef:54:71:00:ce:50:35:58:76:95:08:ee:6a:b1:a2:
                       01:d5
                   Exponent: 65537 (0x10001)
           X509v3 extensions:
               1.3.6.1.4.1.311.21.1:
                   .....
               1.3.6.1.4.1.311.21.2:
                   ....k..wSJ.%7.N.&{. p.
               X509v3 Subject Key Identifier:

13:AD:BF:43:09:BD:82:70:9C:8C:D5:4F:31:6E:D5:22:98:8A:1B:D4
               1.3.6.1.4.1.311.20.2:
                   .
    .S.u.b.C.A
               X509v3 Key Usage:
                   Digital Signature, Certificate Sign, CRL Sign
               X509v3 Basic Constraints: critical
                   CA:TRUE
               X509v3 Authority Key Identifier:

45:66:52:43:E1:7E:58:11:BF:D6:4E:9E:23:55:08:3B:3A:22:6A:A8
               X509v3 CRL Distribution Points:
                   Full Name:

URI:http://crl.microsoft.com/pki/crl/products/MicCorThiParMarRoo_2010-10-05.crl
               Authority Information Access:
                    CA Issuers -
URI:http://www.microsoft.com/pki/certs/MicCorThiParMarRoo_2010-10-05.crt
        Signature Algorithm: sha256WithRSAEncryption
        Signature Value:
            35:08:42:ff:30:cc:ce:f7:76:0c:ad:10:68:58:35:29:46:32:
            76:27:7c:ef:12:41:27:42:1b:4a:aa:6d:81:38:48:59:13:55:
            f3:e9:58:34:a6:16:0b:82:aa:5d:ad:82:da:80:83:41:06:8f:
            b4:1d:f2:03:b9:f3:1a:5d:1b:f1:50:90:f9:b3:55:84:42:28:
            1c:20:bd:b2:ae:51:14:c5:c0:ac:97:95:21:1c:90:db:0f:fc:
            77:9e:95:73:91:88:ca:bd:bd:52:b9:05:50:0d:df:57:9e:a0:
            61:ed:0d:e5:6d:25:d9:40:0f:17:40:c8:ce:a3:4a:c2:4d:af:
            9a:12:1d:08:54:8f:bd:c7:bc:b9:2b:3d:49:2b:1f:32:fc:6a:
            21:69:4f:9b:c8:7e:42:34:fc:36:06:17:8b:8f:20:40:c0:b3:
            9a:25:75:27:cd:c9:03:a3:f6:5d:d1:e7:36:54:7a:b9:50:b5:
            d3:12:d1:07:bf:bb:74:df:dc:1e:8f:80:d5:ed:18:f4:2f:14:
            16:6b:2f:de:66:8c:b0:23:e5:c7:84:d8:ed:ea:c1:33:82:ad:
            56:4b:18:2d:f1:68:95:07:cd:cf:f0:72:f0:ae:bb:dd:86:85:
            98:2c:21:4c:33:2b:f0:0f:4a:f0:68:87:b5:92:55:32:75:a1:
            6a:82:6a:3c:a3:25:11:a4:ed:ad:d7:04:ae:cb:d8:40:59:a0:
            84:d1:95:4c:62:91:22:1a:74:1d:8c:3d:47:0e:44:a6:e4:b0:
            9b:34:35:b1:fa:b6:53:a8:2c:81:ec:a4:05:71:c8:9d:b8:ba:
            e8:1b:44:66:e4:47:54:0e:8e:56:7f:b3:9f:16:98:b2:86:d0:
            68:3e:90:23:b5:2f:5e:8f:50:85:8d:c6:8d:82:5f:41:a1:f4:
            2e:0d:e0:99:d2:6c:75:e4:b6:69:b5:21:86:fa:07:d1:f6:e2:
            4d:d1:da:ad:2c:77:53:1e:25:32:37:c7:6c:52:72:95:86:b0:
            f1:35:61:6a:19:f5:b2:3b:81:50:56:a6:32:2d:fe:a2:89:f9:
            42:86:27:18:55:a1:82:ca:5a:9b:f8:30:98:54:14:a6:47:96:
            25:2f:c8:26:e4:41:94:1a:5c:02:3f:e5:96:e3:85:5b:3c:3e:
            3f:bb:47:16:72:55:e2:25:22:b1:d9:7b:e7:03:06:2a:a3:f7:
            1e:90:46:c3:00:0d:d6:19:89:e3:0e:35:27:62:03:71:15:a6:
            ef:d0:27:a0:a0:59:37:60:f8:38:94:b8:e0:78:70:f8:ba:4c:
            86:87:94:f6:e0:ae:02:45:ee:65:c2:b6:a3:7e:69:16:75:07:
            92:9b:f5:a6:bc:59:83:58

    [key 4]
    SHA1 Fingerprint:
73:8a:96:2b:d9:c8:1b:72:77:17:af:17:ee:09:3f:e9:b4:ba:ee:c0
    Certificate:
        Data:
            Version: 3 (0x2)
            Serial Number:
                e3:4c:a7:5a:0a:61:58:53
            Signature Algorithm: sha256WithRSAEncryption
            Issuer: C=US, ST=California, L=Palo Alto, O=VMware,
Inc., CN=VMware Secure Boot Signing
            Validity
                Not Before: Oct 24 06:47:59 2017 GMT
                Not After : Oct 19 06:47:59 2037 GMT
            Subject: C=US, ST=California, L=Palo Alto, O=VMware,
Inc., CN=VMware Secure Boot Signing
            Subject Public Key Info:
                Public Key Algorithm: rsaEncryption
                    Public-Key: (2048 bit)
                    Modulus:
                        00:e3:71:a4:56:72:10:92:97:49:79:52:5f:91:6f:

71:40:9d:19:f9:fb:78:04:29:bb:ab:e3:c0:af:86:

5d:16:a1:49:d5:ae:eb:f1:dc:6b:07:8e:de:d6:b9:

98:22:6e:bb:01:8f:80:5d:7c:f5:27:ef:ef:7b:9e:

4c:24:80:6a:6a:cb:0e:d4:fe:74:62:1f:67:34:af:
                        8a:25:22:61:f9:53:dc:03:04:ed:99:08:a0:be:78:

b6:a8:86:e9:f3:02:eb:8e:9e:c6:6a:15:43:92:e2:

a1:da:ea:ed:46:10:d3:c8:30:c0:a9:2d:6f:41:5a:

77:75:f8:4b:3d:6a:67:73:dc:e7:19:0e:46:8b:af:
```

```
fa:b1:ec:b6:4f:3d:99:ad:9e:59:e8:da:39:06:26:

13:82:72:bb:d7:16:58:45:da:c0:3b:78:d4:5d:f4:
                        04:46:30:91:fe:71:0d:fb:d3:b7:83:88:4f:81:02:

ce:5c:b5:0b:bd:cc:e9:e9:01:12:1b:f1:68:81:37:
                        60:14:ed:5b:6b:58:59:a0:22:87:33:86:49:65:56:

9e:da:cd:16:45:1f:9a:34:d9:05:00:42:04:17:5a:

c2:27:fa:10:6f:72:65:0b:0c:71:7e:75:5e:6c:90:

10:04:73:ec:e0:a5:2b:6c:c6:1c:c2:cf:19:36:86:
                        c1:1f
                    Exponent: 65537 (0x10001)
            X509v3 extensions:
                X509v3 Subject Key Identifier:

04:59:7F:3E:1F:FB:24:0B:BA:0F:F0:F0:5D:5E:B0:5F:3E:15:F6:D7
                X509v3 Authority Key Identifier:

04:59:7F:3E:1F:FB:24:0B:BA:0F:F0:F0:5D:5E:B0:5F:3E:15:F6:D7
                X509v3 Basic Constraints:
                    CA:TRUE
        Signature Algorithm: sha256WithRSAEncryption
        Signature Value:
            08:bd:b6:a6:bb:55:a3:eb:d8:4c:4c:60:2f:ef:26:d9:7f:37:
            7a:48:2a:9d:b3:94:21:21:3e:07:da:76:7b:ec:37:01:ea:ff:
            05:1e:e6:cf:88:1a:85:4e:b4:1a:ac:85:c2:0c:1e:62:08:8f:
            b3:f9:5a:84:5a:22:e6:f9:1d:c4:d0:30:9e:39:3c:91:15:78:
            f5:e2:20:1b:52:0a:6d:14:61:31:95:53:46:ff:20:19:6f:94:
            04:99:4a:b2:09:4d:82:ad:db:30:f2:7d:26:21:2a:13:5c:c1:
            79:9a:38:c4:29:e3:10:27:8a:05:a1:e0:41:6a:bb:a2:3a:ec:
            84:82:b3:aa:ad:2f:58:dc:ed:bc:03:a1:8b:6e:da:de:6b:49:
            7c:d7:1b:e7:c4:08:46:13:0a:e6:54:33:20:cd:c4:b2:77:f1:
```

```
            ce:c8:b7:0b:e7:8c:c3:3b:7c:9c:d9:f2:85:c8:bc:fc:40:17:
            f0:7c:7f:0f:0a:d9:f3:7a:91:b7:09:95:09:07:2e:78:f6:03:
            23:2c:3e:ee:e9:9a:9f:6d:4b:0e:03:06:f0:c3:ae:69:19:0e:
            80:9d:66:3e:e8:fd:b7:fd:f4:6d:a2:e7:78:83:44:84:e2:ad:
            51:9b:51:cc:e6:f1:cf:0f:10:7d:49:0f:d7:1c:02:c5:75:8e:
            c9:52:47:09

    [key 5]
    SHA1 Fingerprint:
06:60:eb:8d:62:81:35:60:cf:fa:67:e9:75:7c:35:59:5c:d8:9b:f1
    Certificate:
        Data:
            Version: 3 (0x2)
            Serial Number:
                c9:d7:31:83:49:c8:c0:22
            Signature Algorithm: sha256WithRSAEncryption
            Issuer: C=US, ST=California, L=Palo Alto, O=VMware,
Inc., CN=VMware Test Boot Signing
            Validity
                Not Before: Oct 31 10:08:53 2017 GMT
                Not After : Oct 26 10:08:53 2037 GMT
            Subject: C=US, ST=California, L=Palo Alto, O=VMware,
Inc., CN=VMware Test Boot Signing
            Subject Public Key Info:
                Public Key Algorithm: rsaEncryption
                    Public-Key: (2048 bit)
                    Modulus:

00:a0:0e:75:df:bd:82:ac:88:0f:af:69:89:1d:95:
                        9b:28:ec:63:d8:78:e3:0f:67:a0:37:80:84:73:92:

a3:65:ad:1b:ff:d5:d4:28:72:1a:0d:0f:a1:f4:00:

e8:25:fa:bf:a3:f4:e6:c4:bd:5d:c8:0d:a0:82:bf:

e0:5d:c3:a6:f5:fc:dd:5d:36:92:fe:59:15:8a:31:
```

```
1b:68:8d:dd:ab:bf:73:67:18:8f:5b:f9:a9:63:d9:

f4:4c:00:33:ef:9d:13:24:da:64:6c:6b:37:08:fd:

1d:46:b9:d9:2b:25:50:4e:c0:81:97:78:ba:32:b9:

9f:0c:4b:3d:73:f3:f6:00:0f:6d:0b:44:60:a2:3c:

b8:8a:be:55:cf:7b:a5:e0:87:20:7b:0f:f7:ab:7b:

1d:60:71:1d:e3:fc:ba:59:b6:2a:59:64:af:62:b3:

8e:b1:49:cb:60:94:c0:1f:20:7d:c1:26:26:40:3e:

ec:57:94:ba:96:c9:c4:99:fe:ae:36:6a:a0:c4:73:

c4:b9:4b:92:1d:43:5e:2d:ad:1f:f3:09:0d:d6:9e:

cd:26:0e:e1:4a:a7:d0:f6:08:47:94:bc:3a:3b:0f:

f7:23:b5:7d:e6:86:c0:e6:f1:8a:9f:3e:fb:f5:cd:

01:3f:c6:1a:9b:12:2a:12:97:51:cc:ce:d3:06:71:
                        f6:a5
                    Exponent: 65537 (0x10001)
            X509v3 extensions:
                X509v3 Subject Key Identifier:

1C:5B:DA:6D:3A:4B:5B:EB:3B:F9:20:35:62:FC:CB:34:20:D4:E2:3A
                X509v3 Authority Key Identifier:

1C:5B:DA:6D:3A:4B:5B:EB:3B:F9:20:35:62:FC:CB:34:20:D4:E2:3A
                X509v3 Basic Constraints:
                    CA:TRUE
        Signature Algorithm: sha256WithRSAEncryption
        Signature Value:
            9e:a7:02:a4:64:6d:01:cc:0c:96:8e:b4:c5:ee:e1:1d:53:a8:
            9a:1e:10:de:e9:79:54:6c:33:d9:94:42:10:82:51:dc:c5:ce:
            44:23:f0:b7:84:82:d4:46:57:27:de:00:e7:40:37:6d:a9:05:
            5f:e8:c8:eb:d4:8e:9a:e5:ba:87:7e:83:e8:fe:89:4a:a8:32:
            80:4b:17:13:5d:da:b8:0e:69:8e:f8:4a:c6:5f:ae:4b:08:2f:
            87:8e:e0:c0:d3:80:7d:83:89:3b:93:21:19:a3:ce:9e:d8:82:
            d5:3e:61:65:4e:cc:a8:44:11:cb:58:ff:e2:d5:a9:81:06:8e:
            95:ee:96:e4:b1:82:80:7b:54:8e:8e:b7:8b:fe:93:26:31:a5:
            25:35:8b:ac:9d:7b:15:a4:89:38:b2:ed:15:61:db:55:5c:d4:
            02:04:be:d5:6b:fe:9f:cc:0d:67:23:97:b8:d9:41:35:c7:24:
            c4:7a:ef:e4:f3:76:28:34:dc:5f:d6:84:99:39:44:1f:e4:d3:
            de:dd:5e:f4:04:bd:4a:a7:b8:8c:a7:69:4a:12:b1:3b:7c:66:
            28:de:ca:a3:79:98:f0:e7:8f:7f:5a:26:4f:09:d1:48:cf:e2:
            4b:b3:31:b9:21:70:59:d2:95:3a:e1:c8:ac:23:38:79:d0:b9:
            30:21:27:f3

    [key 6]
    SHA1 Fingerprint:
76:a0:92:06:58:00:bf:37:69:01:c3:72:cd:55:a9:0e:1f:de:d2:e0
    Certificate:
        Data:
            Version: 3 (0x2)
            Serial Number:
                b9:41:24:a0:18:2c:92:67
            Signature Algorithm: sha256WithRSAEncryption
            Issuer: C=GB, ST=Isle of Man, L=Douglas, O=Canonical
Ltd., CN=Canonical Ltd. Master Certificate Authority
            Validity
                Not Before: Apr 12 11:12:51 2012 GMT
                Not After : Apr 11 11:12:51 2042 GMT
            Subject: C=GB, ST=Isle of Man, L=Douglas, O=Canonical
Ltd., CN=Canonical Ltd. Master Certificate Authority
            Subject Public Key Info:
                Public Key Algorithm: rsaEncryption
                    Public-Key: (2048 bit)
```

```
                     Modulus:

00:bf:5b:3a:16:74:ee:21:5d:ae:61:ed:9d:56:ac:

bd:de:de:72:f3:dd:7e:2d:4c:62:0f:ac:c0:6d:48:

08:11:cf:8d:8b:fb:61:1f:27:cc:11:6e:d9:55:3d:

39:54:eb:40:3b:b1:bb:e2:85:34:79:ca:f7:7b:bf:

ba:7a:c8:10:2d:19:7d:ad:59:cf:a6:d4:e9:4e:0f:

da:ae:52:ea:4c:9e:90:ce:c6:99:0d:4e:67:65:78:

5d:f9:d1:d5:38:4a:4a:7a:8f:93:9c:7f:1a:a3:85:

db:ce:fa:8b:f7:c2:a2:21:2d:9b:54:41:35:10:57:

13:8d:6c:bc:29:06:50:4a:7e:ea:99:a9:68:a7:3b:

c7:07:1b:32:9e:a0:19:87:0e:79:bb:68:99:2d:7e:

93:52:e5:f6:eb:c9:9b:f9:2b:ed:b8:68:49:bc:d9:

95:50:40:5b:c5:b2:71:aa:eb:5c:57:de:71:f9:40:

0a:dd:5b:ac:1e:84:2d:50:1a:52:d6:e1:f3:6b:6e:

90:64:4f:5b:b4:eb:20:e4:61:10:da:5a:f0:ea:e4:

42:d7:01:c4:fe:21:1f:d9:b9:c0:54:95:42:81:52:

72:1f:49:64:7a:c8:6c:24:f1:08:70:0b:4d:a5:a0:

32:d1:a0:1c:57:a8:4d:e3:af:a5:8e:05:05:3e:10:
                          43:a1
```

```
                    Exponent: 65537 (0x10001)
            X509v3 extensions:
                X509v3 Subject Key Identifier:

AD:91:99:0B:C2:2A:B1:F5:17:04:8C:23:B6:65:5A:26:8E:34:5A:63
                X509v3 Authority Key Identifier:

AD:91:99:0B:C2:2A:B1:F5:17:04:8C:23:B6:65:5A:26:8E:34:5A:63
                X509v3 Basic Constraints: critical
                    CA:TRUE
                X509v3 Key Usage:
                    Digital Signature, Certificate Sign, CRL Sign
                X509v3 CRL Distribution Points:
                    Full Name:
                      URI:http://www.canonical.com/secure-boot-master-ca.crl
        Signature Algorithm: sha256WithRSAEncryption
        Signature Value:
            3f:7d:f6:76:a5:b3:83:b4:2b:7a:d0:6d:52:1a:03:83:c4:12:
            a7:50:9c:47:92:cc:c0:94:77:82:d2:ae:57:b3:99:04:f5:32:
            3a:c6:55:1d:07:db:12:a9:56:fa:d8:d4:76:20:eb:e4:c3:51:
            db:9a:5c:9c:92:3f:18:73:da:94:6a:a1:99:38:8c:a4:88:6d:
            c1:fc:39:71:d0:74:76:16:03:3e:56:23:35:d5:55:47:5b:1a:
            1d:41:c2:d3:12:4c:dc:ff:ae:0a:92:9c:62:0a:17:01:9c:73:
            e0:5e:b1:fd:bc:d6:b5:19:11:7a:7e:cd:3e:03:7e:66:db:5b:
            a8:c9:39:48:51:ff:53:e1:9c:31:53:91:1b:3b:10:75:03:17:
            ba:e6:81:02:80:94:70:4c:46:b7:94:b0:3d:15:cd:1f:8e:02:
            e0:68:02:8f:fb:f9:47:1d:7d:a2:01:c6:07:51:c4:9a:cc:ed:
            dd:cf:a3:5d:ed:92:bb:be:d1:fd:e6:ec:1f:33:51:73:04:be:
            3c:72:b0:7d:08:f8:01:ff:98:7d:cb:9c:e0:69:39:77:25:47:
            71:88:b1:8d:27:a5:2e:a8:f7:3f:5f:80:69:97:3e:a9:f4:99:
            14:db:ce:03:0e:0b:66:c4:1c:6d:bd:b8:27:77:c1:42:94:bd:
            fc:6a:0a:bc
  Forbidden Signatures Database (DBX):
    [key 1]
      [SHA-256]
```

```
80b4d96931bf0d02fd91a61e19d14f1da452e66db2408ca8604d411f92659f0a
f52f83a3fa9cfbd6920f722824dbe4034534d25b8507246b3b957dac6e1bce7a
c5d9d8a186e2c82d09afaa2a6f7f2e73870d3e64f72c4e08ef67796a840f0fbd
363384d14d1f2e0b7815626484c459ad57a318ef4396266048d058c5a19bbf76
1aec84b84b6c65a51220a9be7181965230210d62d6d33c48999c6b295a2b0a06
e6ca68e94146629af03f69c2f86e6bef62f930b37c6fbcc878b78df98c0334e5
c3a99a460da464a057c3586d83cef5f4ae08b7103979ed8932742df0ed530c66
58fb941aef95a25943b3fb5f2510a0df3fe44c58c95e0ab80487297568ab9771
5391c3a2fb112102a6aa1edc25ae77e19f5d6f09cd09eeb2509922bfcd5992ea
d626157e1d6a718bc124ab8da27cbb65072ca03a7b6b257dbdcbbd60f65ef3d1
d063ec28f67eba53f1642dbf7dff33c6a32add869f6013fe162e2c32f1cbe56d
29c6eb52b43c3aa18b2cd8ed6ea8607cef3cfae1bafe1165755cf2e614844a44
90fbe70e69d633408d3e170c6832dbb2d209e0272527dfb63d49d29572a6f44c
075eea060589548ba060b2feed10da3c20c7fe9b17cd026b94e8a683b8115238
07e6c6a858646fb1efc67903fe28b116011f2367fe92e6be2b36999eff39d09e
09df5f4e511208ec78b96d12d08125fdb603868de39f6f72927852599b659c26
0bbb4392daac7ab89b30a4ac657531b97bfaab04f90b0dafe5f9b6eb90a06374
0c189339762df336ab3dd006a463df715a39cfb0f492465c600e6c6bd7bd898c
```

```
0d0dbeca6f29eca06f331a7d72e4884b12097fb348983a2a14a0d73f4f10140f

0dc9f3fb99962148c3ca833632758d3ed4fc8d0b0007b95b31e6528f2acd5bfc

106faceacfecfd4e303b74f480a08098e2d0802b936f8ec774ce21f31686689c

174e3a0b5b43c6a607bbd3404f05341e3dcf396267ce94f8b50e2e23a9da920c

18333429ff0562ed9f97033e1148dceee52dbe2e496d5410b5cfd6c864d2d10f

2b99cf26422e92fe365fbf4bc30d27086c9ee14b7a6fff44fb2f6b9001699939

2bbf2ca7b8f1d91f27ee52b6fb2a5dd049b85a2b9b529c5d6662068104b055f8

2c73d93325ba6dcbe589d4a4c63c5b935559ef92fbf050ed50c4e2085206f17d

2e70916786a6f773511fa7181fab0f1d70b557c6322ea923b2a8d3b92b51af7d

306628fa5477305728ba4a467de7d0387a54f569d3769fce5e75ec89d28d1593

3608edbaf5ad0f41a414a1777abf2faf5e670334675ec3995e6935829e0caad2

3841d221368d1583d75c0a02e62160394d6c4e0a6760b6f607b90362bc855b02

3fce9b9fdf3ef09d5452b0f95ee481c2b7f06d743a737971558e70136ace3e73

4397daca839e7f63077cb50c92df43bc2d2fb2a8f59f26fc7a0e4bd4d9751692

47cc086127e2069a86e03a6bef2cd410f8c55a6d6bdb362168c31b2ce32a5adf

518831fe7382b514d03e15c621228b8ab65479bd0cbfa3c5c1d0f48d9c306135

5ae949ea8855eb93e439dbc65bda2e42852c2fdf6789fa146736e3c3410f2b5c

6b1d138078e4418aa68deb7bb35e066092cf479eeb8ce4cd12e7d072ccb42f66
```

```
6c8854478dd559e29351b826c06cb8bfef2b94ad3538358772d193f82ed1ca11
6f1428ff71c9db0ed5af1f2e7bbfcbab647cc265ddf5b293cdb626f50a3a785e
71f2906fd222497e54a34662ab2497fcc81020770ff51368e9e3d9bfcbfd6375
726b3eb654046a30f3f83d9b96ce03f670e9a806d1708a0371e62dc49d2c23c1
72e0bd1867cf5d9d56ab158adf3bddbc82bf32a8d8aa1d8c5e2f6df29428d6d8
7827af99362cfaf0717dade4b1bfe0438ad171c15addc248b75bf8caa44bb2c5
81a8b965bb84d3876b9429a95481cc955318cfaa1412d808c8a33bfd33fff0e4
82db3bceb4f60843ce9d97c3d187cd9b5941cd3de8100e586f2bda5637575f67
895a9785f617ca1d7ed44fc1a1470b71f3f1223862d9ff9dcc3ae2df92163daf
8ad64859f195b5f58dafaa940b6a6167acd67a886e8f469364177221c55945b9
8bf434b49e00ccf71502a2cd900865cb01ec3b3da03c35be505fdf7bd563f521
8d8ea289cfe70a1c07ab7365cb28ee51edd33cf2506de888fbadd60ebf80481c
9998d363c491be16bd74ba10b94d9291001611736fdca643a36664bc0f315a42
9e4a69173161682e55fde8fef560eb88ec1ffedcaf04001f66c0caf707b2b734
a6b5151f3655d3a2af0d472759796be4a4200e5495a7d869754c4848857408a7
a7f32f508d4eb0fead9a087ef94ed1ba0aec5de6f7ef6ff0a62b93bedf5d458d
ad6826e1946d26d3eaf3685c88d97d85de3b4dcb3d0ee2ae81c70560d13c5720
aeebae3151271273ed95aa2e671139ed31a98567303a332298f83709a9d55aa1
```

```
afe2030afb7d2cda13f9fa333a02e34f6751afec11b010dbcd441fdf4c4002b3
b54f1ee636631fad68058d3b0937031ac1b90ccb17062a391cca68afdbe40d55
b8f078d983a24ac433216393883514cd932c33af18e7dd70884c8235f4275736
b97a0889059c035ff1d54b6db53b11b9766668d9f955247c028b2837d7a04cd9
bc87a668e81966489cb508ee805183c19e6acd24cf17799ca062d2e384da0ea7
c409bdac4775add8db92aa22b5b718fb8c94a1462c1fe9a416b95d8a3388c2fc
c617c1a8b1ee2a811c28b5a81b4c83d7c98b5b0c27281d610207ebe692c2967f
c90f336617b8e7f983975413c997f10b73eb267fd8a10cb9e3bdbfc667abdb8b
cb6b858b40d3a098765815b592c1514a49604fafd60819da88d7a76e9778fef7
ce3bfabe59d67ce8ac8dfd4a16f7c43ef9c224513fbc655957d735fa29f540ce
d8cbeb9735f5672b367e4f96cdc74969615d17074ae96c724d42ce0216f8f3fa
e92c22eb3b5642d65c1ec2caf247d2594738eebb7fb3841a44956f59e2b0d1fa
fddd6e3d29ea84c7743dad4a1bdbc700b5fec1b391f932409086acc71dd6dbd8
fe63a84f782cc9d3fcf2ccf9fc11fbd03760878758d26285ed12669bdc6e6d01
fecfb232d12e994b6d485d2c7167728aa5525984ad5ca61e7516221f079a1436
ca171d614a8d7e121c93948cd0fe55d39981f9d11aa96e03450a415227c2c65b
55b99b0de53dbcfe485aa9c737cf3fb616ef3d91fab599aa7cab19eda763b5ba
77dd190fa30d88ff5e3b011a0ae61e6209780c130b535ecb87e6f0888a0b6b2f
```

```
c83cb13922ad99f560744675dd37cc94dcad5a1fcba6472fee341171d939e884

3b0287533e0cc3d0ec1aa823cbf0a941aad8721579d1c499802dd1c3a636b8a9

939aeef4f5fa51e23340c3f2e49048ce8872526afdf752c3a7f3a3f2bc9f6049

64575bd912789a2e14ad56f6341f52af6bf80cf94400785975e9f04e2d64d745

45c7c8ae750acfbb48fc37527d6412dd644daed8913ccd8a24c94d856967df8e
  Machine Owner Keys (MOK):
    [key 1]
    SHA1 Fingerprint:
76:a0:92:06:58:00:bf:37:69:01:c3:72:cd:55:a9:0e:1f:de:d2:e0
    Certificate:
        Data:
            Version: 3 (0x2)
            Serial Number:
                b9:41:24:a0:18:2c:92:67
            Signature Algorithm: sha256WithRSAEncryption
            Issuer: C=GB, ST=Isle of Man, L=Douglas, O=Canonical
Ltd., CN=Canonical Ltd. Master Certificate Authority
            Validity
                Not Before: Apr 12 11:12:51 2012 GMT
                Not After : Apr 11 11:12:51 2042 GMT
            Subject: C=GB, ST=Isle of Man, L=Douglas, O=Canonical
Ltd., CN=Canonical Ltd. Master Certificate Authority
            Subject Public Key Info:
                Public Key Algorithm: rsaEncryption
                    Public-Key: (2048 bit)
                    Modulus:

00:bf:5b:3a:16:74:ee:21:5d:ae:61:ed:9d:56:ac:

bd:de:de:72:f3:dd:7e:2d:4c:62:0f:ac:c0:6d:48:
```

```
08:11:cf:8d:8b:fb:61:1f:27:cc:11:6e:d9:55:3d:

39:54:eb:40:3b:b1:bb:e2:85:34:79:ca:f7:7b:bf:

ba:7a:c8:10:2d:19:7d:ad:59:cf:a6:d4:e9:4e:0f:

da:ae:52:ea:4c:9e:90:ce:c6:99:0d:4e:67:65:78:

5d:f9:d1:d5:38:4a:4a:7a:8f:93:9c:7f:1a:a3:85:

db:ce:fa:8b:f7:c2:a2:21:2d:9b:54:41:35:10:57:

13:8d:6c:bc:29:06:50:4a:7e:ea:99:a9:68:a7:3b:

c7:07:1b:32:9e:a0:19:87:0e:79:bb:68:99:2d:7e:

93:52:e5:f6:eb:c9:9b:f9:2b:ed:b8:68:49:bc:d9:

95:50:40:5b:c5:b2:71:aa:eb:5c:57:de:71:f9:40:

0a:dd:5b:ac:1e:84:2d:50:1a:52:d6:e1:f3:6b:6e:

90:64:4f:5b:b4:eb:20:e4:61:10:da:5a:f0:ea:e4:

42:d7:01:c4:fe:21:1f:d9:b9:c0:54:95:42:81:52:

72:1f:49:64:7a:c8:6c:24:f1:08:70:0b:4d:a5:a0:

32:d1:a0:1c:57:a8:4d:e3:af:a5:8e:05:05:3e:10:
                        43:a1
                    Exponent: 65537 (0x10001)
            X509v3 extensions:
                X509v3 Subject Key Identifier:

AD:91:99:0B:C2:2A:B1:F5:17:04:8C:23:B6:65:5A:26:8E:34:5A:63
```

```
                X509v3 Authority Key Identifier:

AD:91:99:0B:C2:2A:B1:F5:17:04:8C:23:B6:65:5A:26:8E:34:5A:63
                X509v3 Basic Constraints: critical
                    CA:TRUE
                X509v3 Key Usage:
                    Digital Signature, Certificate Sign, CRL Sign
                X509v3 CRL Distribution Points:
                    Full Name:
                      URI:http://www.canonical.com/secure-boot-master-ca.crl
        Signature Algorithm: sha256WithRSAEncryption
        Signature Value:
            3f:7d:f6:76:a5:b3:83:b4:2b:7a:d0:6d:52:1a:03:83:c4:12:
            a7:50:9c:47:92:cc:c0:94:77:82:d2:ae:57:b3:99:04:f5:32:
            3a:c6:55:1d:07:db:12:a9:56:fa:d8:d4:76:20:eb:e4:c3:51:
            db:9a:5c:9c:92:3f:18:73:da:94:6a:a1:99:38:8c:a4:88:6d:
            c1:fc:39:71:d0:74:76:16:03:3e:56:23:35:d5:55:47:5b:1a:
            1d:41:c2:d3:12:4c:dc:ff:ae:0a:92:9c:62:0a:17:01:9c:73:
            e0:5e:b1:fd:bc:d6:b5:19:11:7a:7e:cd:3e:03:7e:66:db:5b:
            a8:c9:39:48:51:ff:53:e1:9c:31:53:91:1b:3b:10:75:03:17:
            ba:e6:81:02:80:94:70:4c:46:b7:94:b0:3d:15:cd:1f:8e:02:
            e0:68:02:8f:fb:f9:47:1d:7d:a2:01:c6:07:51:c4:9a:cc:ed:
            dd:cf:a3:5d:ed:92:bb:be:d1:fd:e6:ec:1f:33:51:73:04:be:
            3c:72:b0:7d:08:f8:01:ff:98:7d:cb:9c:e0:69:39:77:25:47:
            71:88:b1:8d:27:a5:2e:a8:f7:3f:5f:80:69:97:3e:a9:f4:99:
            14:db:ce:03:0e:0b:66:c4:1c:6d:bd:b8:27:77:c1:42:94:bd:
            fc:6a:0a:bc
```

Deploying NVIDIA Converged Accelerator

Info

It is recommended to upgrade your BlueField product to the latest software and firmware versions available to benefit from new features and latest bug fixes.

This section assumes that you have installed the BlueField OS BFB on your NVIDIA® Converged Accelerator using any of the following guides:

- Deploying BlueField Software Using BFB from Host
- Deploying BlueField Software from BlueField BMC
- Deploying BlueField Software Using PXE

NVIDIA® CUDA® (GPU driver) must be installed to use the GPU. For information on how to install CUDA on your Converged Accelerator, refer to NVIDIA CUDA Installation Guide for Linux.

Configuring Operation Mode

After installing the BFB, you may now select the mode you want your NVIDIA Converged Accelerator to operate in.

- Standard (default) – the NVIDIA® BlueField® and the GPU operate separately (GPU is owned by the host)
- BlueField-X – the GPU is exposed to BlueField and is no longer visible on the host (GPU is owned by BlueField)

Note

It is important to know your device name (e.g., `mt41686_pciconf0`).

MST tool is necessary for this purpose which is installed by default on the DPU.

Run:

```
mst status -v
```

Example output:

```
MST modules:
------------
    MST PCI module is not loaded
    MST PCI configuration module loaded
PCI devices:
------------
DEVICE_TYPE             MST
PCI       RDMA            NET
NUMA
BlueField2(rev:1)
/dev/mst/mt41686_pciconf0.1   3b:00.1   mlx5_1
net-ens1f1                0

BlueField2(rev:1)       /dev/mst/mt41686_pciconf0
3b:00.0   mlx5_0          net-ens1f0
0
```

BlueField-X Mode

1. Run the following command from the host:

   ```
   mlxconfig -d /dev/mst/<device-name> s
   PCI_DOWNSTREAM_PORT_OWNER[4]=0xF
   ```

2. P erform a BlueField system-level reset for the `mlxconfig` settings to take effect.

Standard Mode

To return BlueField from BlueField-X mode to Standard mode:

1. Run the following command from the host:

   ```
   mlxconfig -d /dev/mst/<device-name> s
   PCI_DOWNSTREAM_PORT_OWNER[4]=0x0
   ```

2. P erform a BlueField system-level reset for the `mlxconfig` settings to take effect.

Verifying Configured Operational Mode

Use the following command from the host or BlueField:

```
$ sudo mlxconfig -d /dev/mst/<device-name> q
PCI_DOWNSTREAM_PORT_OWNER[4]
```

- Example of Standard mode output:

  ```
  Device #1:
  ```

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

[...]

Configurations:                                  Next Boot
         PCI_DOWNSTREAM_PORT_OWNER[4]
DEVICE_DEFAULT(0)
```

- Example of BlueField-X mode output:

```
Device #1:
----------
[...]

Configurations:                                  Next Boot
         PCI_DOWNSTREAM_PORT_OWNER[4]        EMBEDDED_CPU(15)
```

Verifying GPU Ownership

The following are example outputs for when BlueField is configured to BlueField-X mode.

The GPU is no longer visible from the host:

```
root@host:~# lspci | grep -i nv
None
```

The GPU is now visible from BlueField:

```
ubuntu@bf:~$ lspci | grep -i nv
06:00.0 3D controller: NVIDIA Corporation GA20B8 (rev a1)
```

GPU Firmware

Get GPU Firmware

```
smbpbi: (See SMBPBI spec)

root@bf:~# i2cset -y 3 0x4f 0x5c 0x05 0x08 0x00 0x80 s
root@bf:~# i2cget -y 3 0x4f 0x5c ip 5
5: 0x04 0x05 0x08 0x00 0x5f
root@bf:~# i2cget -y 3 0x4f 0x5d ip 5
5: 0x04 0x39 0x32 0x2e 0x30
root@bf:~#
root@bf:~#
root@bf:~# i2cset -y 3 0x4f 0x5c 0x05 0x08 0x01 0x80 s
root@bf:~# i2cget -y 3 0x4f 0x5c ip 5
5: 0x04 0x05 0x08 0x01 0x5f
root@bf:~# i2cget -y 3 0x4f 0x5d ip 5
5: 0x04 0x30 0x2e 0x36 0x42
root@bf:~# i2cset -y 3 0x4f 0x5c 0x05 0x08 0x02 0x80 s
root@bf:~# i2cget -y 3 0x4f 0x5c ip 5
5: 0x04 0x05 0x08 0x02 0x5f
root@bf:~# i2cget -y 3 0x4f 0x5d ip 5
5: 0x04 0x2e 0x30 0x30 0x2e
root@bf:~# i2cset -y 3 0x4f 0x5c 0x05 0x08 0x03 0x80 s
root@bf:~# i2cget -y 3 0x4f 0x5c ip 5
5: 0x04 0x05 0x08 0x03 0x5f
root@bf:~# i2cget -y 3 0x4f 0x5d ip 5
5: 0x04 0x30 0x31 0x00 0x00
root@bf:~#

39 32 2e 30 30 2e 36 42 2e 30 30 2e 30 31 00 00   92.00.6B.00.01
```

Updating GPU Firmware

```
root@bf:~# scp root@10.23.201.227:/<path-to-fw-
bin>/1004_0230_891__92006B0001-dbg-ota.bin /tmp/gpu_images/
root@10.23.201.227's password:
1004_0230_891__92006B0001-dbg-ota.bin
100%  384KB 384.4KB/s   00:01

root@bf:~# cat /tmp/gpu_images/progress.txt
TaskState="Running"
TaskStatus="OK"
TaskProgress="50"

root@bf:~# cat /tmp/gpu_images/progress.txt
TaskState="Running"
TaskStatus="OK"
TaskProgress="50"

root@bf:~# cat /tmp/gpu_images/progress.txt
TaskState=Frimware update succeeded.
TaskStatus=OK
TaskProgress=100
```

Installing Repo Package on Host Side

ⓘ **Note**

This section assumes that an NVIDIA® BlueField® networking platform (DPU or SuperNIC) has already been installed in a server according to the instructions detailed in the BlueField's hardware user guide.

To install the repository packages on the host side, follow the instructions detailed in the following sections of the *NVIDIA DOCA Installation Guide for Linux*:

1. Uninstalling Software from Host
2. Installing Prerequisites on Host for Target BlueField
3. GPG and Kernel Module Signing (and all relevant subsections)
4. Installing Software on Host
    1. DOCA Extra Package and doca-kernel-support

Installing Popular Linux Distributions on BlueField

Building Your Own BFB Installation Image

Users wishing to build their own customized NVIDIA® BlueField® networking platform's (DPU or SuperNIC) OS image can use the BFB build environment. See this GitHub webpage for more information.

ⓘ **Note**

For any customized BlueField OS image to boot on the UEFI secure-boot-enabled BlueField (default BlueField secure boot setting), the OS must be either signed with an existing key in the UEFI DB (e.g., the Microsoft key), or UEFI secure boot must be disabled. See "Secure Boot" and its subpages for more details.

Installing Linux Distributions

Contact NVIDIA Enterprise Support for information on the installation of Linux distributions other than Ubuntu.

BlueField Linux Drivers

The following table lists the BlueField drivers which are part of the Official Ubuntu Linux distribution for BlueField. Some of the drivers are not in the upstream Linux kernel yet.

Driver	Description	BlueField-2	BlueField-3
`bluefield-edac`	BlueField-specific EDAC driver	☐	☐
`dw_mmc_bluefield`	BlueField DW Multimedia Card driver	☐	☐
`sdhci-of-dwcmshc`	SDHCI platform driver for Synopsys DWC MSHC	☐	☐
`gpio-mlxbf2`	GPIO driver	☐	☐
`gpio-mlxbf3`	GPIO driver	☐	☐
`i2c-mlx`	I2C bus driver (`i2c-mlxbf.c` upstream)	☐	☐
`ipmb-dev-int`	Driver needed to receive IPMB messages from a BMC and send a response back. This driver works with the I2C driver and a user-space program such as OpenIPMI.	☐	☐
`ipmb-host`	Driver needed on BlueField to send IPMB messages to the BMC on the IPMB bus. This driver works with the I2C driver. It only loads successfully if it executes a successful handshake with the BMC.	☐	☐

Driver	Description	BlueField-2	BlueField-3
`mlxbf-gige`	Gigabit Ethernet driver	▯	▯
`mlxbf-livefish`	BlueField HCA firmware burning driver. This driver supports burning firmware for the embedded HCA in the BlueField SoC.	▯	▯
`mlxbf-pka`	BlueField PKA kernel module	▯	▯
`mlxbf-pmc`	Performance monitoring counters. The driver provides access to available performance modules through the `sysfs` interface. The performance modules in BlueField are present in several hardware blocks and each block has a certain set of supported events.	▯	▯
`mlxbf-ptm`	Kernel driver that provides a debufgs interface for the system software to monitor the BlueField device's power and thermal management parameters.	▯	▯
`mlxbf-tmfifo`	TMFIFO driver for BlueField SoC	▯	▯
`mlx-bootctl`	Boot control driver. This driver provides a `sysfs` interface for systems management software to manage reset time actions.	▯	▯
`mlx-trio`	TRIO driver for BlueField SoC	▯	▯
`pwr-mlxbf`	Supports reset or low-power mode handling for BlueField.	▯	▯

Driver	Description	BlueField-2	BlueField-3
`pinctrl-mlxbf`	Allows multiplexing individual GPIOs to switch from the default hardware mode to software-controlled mode.	▯	▯
`mlxbf-pmc`	Mellanox PMC driver	▯	▯

Updating BlueField Software Packages Using Standard Linux Tools

Please refer to section "Upgrading BlueField Using Standard Linux Tools" of the *NVIDIA DOCA Installation Guide for Linux* for information.

Upgrading BlueField Software Components Using PLDM

The PLDM firmware update is a standardized protocol that enables out-of-band (OOO) firmware upgrades for devices by transferring firmware images between an update agent (e.g., the server's platform BMC) and target devices (i.e., NVIDIA® BlueField®-3). BlueField-3 firmware components can be upgraded using this method.

The BlueField-3 PLDM firmware image includes the following components: NIC firmware, ATF/UEFI, BMC firmware, and CEC firmware (i.e., does not include Arm OS or DOCA software). The BlueField-3 PLDM image is specific to each BlueField-3 SKU and can be downloaded from the NVIDIA DOCA Downloads page.

Info

PLDM firmware update is supported in both NIC and DPU modes of operation.

In DPU mode, Linux runs on BlueField's Arm OS. The PLDM firmware update is handled by the `/etc/acpi/actions/bf-upgrade` script, which is triggered by an ACPI event.

Note

In DPU mode, the PLDM firmware update requires the relevant credentials to upgrade the BlueField BMC and CEC firmware. To provide access to the BlueField BMC, users must update the `/etc/bf-upgrade.conf` local Arm OS file with the BlueField BMC credentials. The format for `/etc/bf-upgrade.conf` is the same as `bf.cfg`.

00000195-7ade-df33-a3dd-7fff58990000

Warning

PLDM firmware update is possible only if the currently running version (i.e., the version to update from) is DOCA 2.9.0/BSP 4.9.0 or higher.

Applying BlueField PLDM Image

Note, server reboot will not restart the BlueField BMC, (BMC, CEC not applied).

Info

The PLDM file can be downloaded from the NVIDIA DOCA Downloads page.

The BlueField must be restarted to apply the new firmware images. To restart BlueField:

1. Gracefully shut down the BlueField Arm OS.
2. Perform a server power cycle.

Alternatively, you can reboot the server instead of performing a power cycle using the following steps:

1. Gracefully shut down the BlueField Arm OS and wait until it completes.

    Note

> Without a graceful shutdown of the BlueField Arm OS during a server reboot, the BlueField Arm side does not undergo a restart process (only the NIC firmware is applied).

2. Reboot the server (this applies ATF, UEFI, Arm OS, and NIC firmware).
3. Log into the BlueField BMC via Redfish and issue a restart (this applies the BlueField BMC and CEC firmware).

> **Note**
>
> A server reboot does not restart the BlueField BMC, so the BMC and CEC firmware is not applied.