Troubleshooting and How-Tos

Table of contents

NVIDIA BlueField Reset and Reboot Procedures

BlueField System Reboot

This section describes the necessary operations to load new NIC firmware, following NVIDIA® BlueField® NIC firmware update. This procedure deprecates the need for full server power cycle.

The following steps are executed in the BlueField OS:

1. Issue a query command to ascertain whether BlueField system reboot is supported by your environment:

   ```
   mlxfwreset -d 03:00.0 q
   ```

   If the output includes the following lines, proceed to step 2:

   ```
   3: Driver restart and PCI reset                    -Supported (default)
   ...
   1: Driver is the owner                             -Supported (default)
   ```

   Note

   If it says `Not Supported` instead, then proceed to the instructions under section "BlueField System-level Reset".

2. Issue a BlueField system reboot:

```
mlxfwreset -d 03:00.0 -y -l 3 --sync 1 r
```

BlueField System-level Reset

This section describes the way to perform system-level reset (SLR) which is necessary for firmware configuration changes to take effect.

- SLR for BlueField running in DPU mode
- SLR for BlueField running in NIC mode
- SLR for BlueField running in DPU mode on hosts with separate power control (special use case)

System-level Reset for BlueField in DPU Mode

The following is the high-level flow of the procedure:

1. Graceful shutdown of BlueField Arm cores.
2. Query BlueField state to affirm shutdown reached.

> **Info**
>
> In systems with multiple BlueField networking platforms, repeat steps 1 and 2 for all devices before proceeding.

3. Warm reboot the server.

Step by step process:

Info

Some of the following steps can be performed using different methods, depending on resource availability and support in the user's environment.

1. Graceful shutdown of BlueField Arm cores.

   Info

   This operation is expected to finish within 15 seconds.

   Possible methods:

   - From the BlueField OS:

     ```
     shutdown -h now
     ```

     Or:

     ```
     mlxfwreset -d /dev/mst/mt*pciconf0 -l 1 -t 4 --sync 0 r
     ```

   - From the host OS:

     Info

Not relevant when the BlueField is operating in Zero-Trust Mode.

```
mlxfwreset -d <mst-device> -l 1 -t 4 r
```

- Using the BlueField BMC:

  ```
  ipmitool -C 17 -I lanplus -H <bmc_ip> -U root -P <password> power soft
  ```

  Or using Redfish (BlueField-3 and above):

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

2. Query BlueField state.

   Possible methods:

   - From the host OS:

     Info

     Not relevant when the BlueField is operating in Zero-Trust Mode.

     ```
     echo DISPLAY_LEVEL 2 > /dev/rshim0/misc
     ```

```
cat /dev/rshim0/misc
```

Expected output:

```
INFO[BL31]: System Off
```

- Utilizing the BlueField BMC:

  ```
  ipmitool -C 17 -I lanplus -H <bmc_ip> -U root -P <password> raw 0x32 0xA3
  ```

  Expected output: `06`.

3. Warm reboot the server f rom the host OS:

```
mlxfwreset -d <mst-device> -l 4 r
```

Note

If multiple BlueField devices are present in the host, this command must run only once. In this case, the MST device can be of any of the BlueFields for which the reset is necessary and participated in step 1.

Or:

```
reboot
```

Note

For external hosts which do not toggle PERST# in their standard reboot command, use the mlxfwreset option.

System-level Reset for BlueField in NIC Mode

Perform warm reboot of the host OS:

```
mlxfwreset -d <mst-device> -l 4 r
```

Or:

```
reboot
```

Note

For external hosts which do not toggle PERST# in their standard reboot command, use the mlxfwreset option.

System-level Reset for Host with Separate Power Control

This procedure is a special use case relevant only to host platforms with separate power control for the PCIe slot and CPUs, in which the BlueField (running in DPU mode) is provided power while host OS/CPUs may be in shutdown or similar standby state (this allows the BlueField device to be operational while the host CPU is in shutdown/standby state).

The following is the high-level flow of the procedure:

1. Graceful shutdown of host OS or similar CPU standby.
2. Graceful shutdown of BlueField Arm cores.
3. Query BlueField state to affirm shutdown reached.
4. Full BlueField Reset
5. Query BlueField state to affirm operational state reached

> **Info**
>
> In systems with multiple BlueField networking platforms, repeat steps 1 through 5 for all devices before proceeding.

6. Power on the server.

Step by step process:

> **Info**
>
> Some of the following steps can be performed using different methods, depending on resource availability and support in the user's environment.

1. Graceful shutdown of host OS by any means preferable.
2. Graceful shutdown of BlueField Arm cores.

Info

This step normally takes up to 15 seconds to complete.

- From the BlueField OS:

  ```
  shutdown -h now
  ```

- Utilizing the BlueField BMC:
  - Using IPMI:

    ```
    ipmitool -C 17 -I lanplus -H <bmc_ip> -U root -P <password> power soft
    ```

  - Using Redfish (for BlueField-3 and above):

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

3. Query the BlueField's state utilizing the BlueField BMC:

   ```
   ipmitool -C 17 -I lanplus -H <bmc_ip> -U root -P <password> raw 0x32 0xA3
   ```

   Expected output: `06`.

4. Perform BlueField hard reset utilizing the BlueField BMC:

   Info

This step takes up to 2 minutes to complete .

- Using IPMI:

```
ipmitool -C 17 -I lanplus -H <bmc_ip> -U root -P <password> power cycle
```

- Using Redfish (for BlueField-3 and above):

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

5. Query BlueField operational state u tilizing the BlueField BMC :

Info

At this point, the BlueField is expected to b e operational .

```
ipmitool -C 17 -I lanplus -H <bmc_ip> -U root -P <password> raw 0x32 0xA3
```

Expected output: `05`.

6. Power on/boot up the host OS.

RShim Troubleshooting and How-Tos

Another backend already attached

Several generations of NVIDIA® BlueField® networking platforms (DPUs or SuperNICs) are equipped with a USB interface in which RShim can be routed, via USB cable, to an external host running Linux and the RShim driver.

In this case, typically following a system reboot, the RShim over USB prevails and the BlueField host reports RShim status as "another backend already attached". This is correct behavior, since there can only be one RShim backend active at any given time. However, this means that the BlueField host does not own RShim access.

To reclaim RShim ownership safely:

1. Stop the RShim driver on the remote Linux. Run:

```
systemctl stop rshim
systemctl disable rshim
```

2. Restart RShim on the BlueField host. Run:

```
systemctl enable rshim
systemctl start rshim
```

The "another backend already attached" scenario can also be attributed to the RShim backend being owned by the BMC in BlueField devices with integrated BMC. This is elaborated on further down on this page.

RShim driver not loading

Verify whether your BlueField features an integrated BMC or not. Run:

```
# sudo sudo lspci -s $(sudo lspci -d 15b3: | head -1 | awk '{print $1}') -vvv | grep "Product Name"
```

Example output for BlueField with an integrated BMC:

```
Product Name: BlueField-2 DPU 25GbE Dual-Port SFP56, integrated BMC, Crypto and Secure Boot
Enabled, 16GB on-board DDR, 1GbE OOB management, Tall Bracket, FHHL
```

If your BlueField has an integrated BMC, refer to RShim driver not loading on host with integrated BMC.

If your BlueField does not have an integrated BMC, refer to RShim driver not loading on host on BlueField without integrated BMC.

RShim driver not loading on BlueField with integrated BMC

RShim driver not loading on host

1. Access the BMC via the RJ45 management port of BlueField.
2. Delete RShim on the BMC:

   ```
   systemctl stop rshim
   systemctl disable rshim
   ```

3. Enable RShim on the host:

   ```
   systemctl enable rshim
   systemctl start rshim
   ```

4. Restart RShim service. Run:

```
sudo systemctl restart rshim
```

If RShim service does not launch automatically, run:

```
sudo systemctl status rshim
```

This command is expected to display "active (running)".

5. Display the current setting. Run:

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

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

RShim driver not loading on BMC

1. Verify that the RShim service is not running on host. Run:

```
systemctl status rshim
```

If the output is active, then it may be presumed that the host has ownership of the RShim.

2. Delete RShim on the host. Run:

```
systemctl stop rshim
systemctl disable rshim
```

3. Enable RShim on the BMC. Run:

```
systemctl enable rshim
systemctl start rshim
```

4. Display the current setting. Run:

```
# cat /dev/rshim<N>/misc | grep DEV_NAME
DEV_NAME        usb-1.0
```

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

RShim driver not loading on host on BlueField without integrated BMC

1. Download the suitable DEB/RPM for RShim (management interface for BlueField from the host) driver.
2. Reinstall RShim package on the host.
   - For Ubuntu/Debian, run:

     ```
     sudo dpkg --force-all -i rshim-<version>.deb
     ```

   - For RHEL/CentOS, run:

     ```
     sudo rpm -Uhv rshim-<version>.rpm
     ```

3. Restart RShim service. Run:

```
sudo systemctl restart rshim
```

If RShim service does not launch automatically, run:

```
sudo systemctl status rshim
```

This command is expected to display "active (running)".

4. Display the current setting. Run:

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

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

Change ownership of RShim from NIC BMC to host

1. Verify that your card has BMC. Run the following on the host:

```
# sudo sudo lspci -s $(sudo lspci -d 15b3: | head -1 | awk '{print $1}') -vvv |grep "Product Name"
Product Name: BlueField-2 DPU 25GbE Dual-Port SFP56, integrated BMC, Crypto and Secure
Boot Enabled, 16GB on-board DDR, 1GbE OOB management, Tall Bracket, FHHL
```

The product name is supposed to show "integrated BMC" .

2. Access the BMC via the RJ45 management port of BlueField.

3. Delete RShim on the BMC:

```
systemctl stop rshim
systemctl disable rshim
```

4. Enable RShim on the host:

```
systemctl enable rshim
systemctl start rshim
```

5. Restart RShim service. Run:

```
sudo systemctl restart rshim
```

   If RShim service does not launch automatically, run:

```
sudo systemctl status rshim
```

   This command is expected to display "`active (running)`".

6. Display the current setting. Run:

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

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

How to support multiple BlueField devices on the host

For more information, refer to section "RShim Multiple Board Support".

BFB installation monitoring

The BFB installation flow can be traced using various interfaces:

- From the host:
  - RShim console (/dev/rshim0/console)
  - RShim log buffer (/dev/rshim0/misc); also included in bfb-install's output
  - UART console (/dev/ttyUSB0)
- From the BMC console:
  - SSH to the BMC and run obmc-console-client

    ⓘ **Info**

    Additional information about BMC interfaces is available in BMC software documentation

- From the BlueField:
  - /root/<OS>.installation.log available on the BlueField Arm OS after installation

Connectivity Troubleshooting

Connection (ssh, screen console) to the BlueField is lost

The UART cable in the Accessories Kit (OPN: MBF20-DKIT) can be used to connect to the NVIDIA® BlueField® networking platform (DPU or SuperNIC) console and identify the stage at which BlueField is hanging.

Follow this procedure:

1. Connect the UART cable to a USB socket, and find it in your USB devices.

```
sudo lsusb
Bus 002 Device 003: ID 0403:6001 Future Technology Devices International, Ltd FT232 Serial
(UART) IC
```

Note

For more information on the UART connectivity, please refer to the BlueField's hardware user guide under Supported Interfaces > Interfaces Detailed Description > NC-SI Management Interface.

Info

It is good practice to connect the other end of the NC-SI cable to a different host than the one on which BlueField is installed.

2. Install the minicom application.

   - For CentOS/RHEL:

     ```
     sudo yum install minicom -y
     ```

   - For Ubuntu/Debian:

     ```
     sudo apt-get install minicom
     ```

3. Open the minicom application.

   ```
   sudo minicom -s -c on
   ```

4. Go to "Serial port setup"

5. Enter "F" to change "Hardware Flow control" to NO

6. Enter "A" and change to `/dev/ttyUSB0` and press Enter

7. Press ESC.

8. Type on "Save setup as dfl"

9. Exit minicom by pressing Ctrl + a + z.

   ```
   +-----------------------------------------------------------------------+
   | A -    Serial Device      : /dev/ttyUSB0                           |
   |                                                            |
   | C -   Callin Program      :                                    |
   ```

```
| D -  Callout Program      :                              |
| E -    Bps/Par/Bits       : 115200 8N1                   |
| F - Hardware Flow Control : No                           |
| G - Software Flow Control : No                           |
|                                                          |
|    Change which setting?                                 |
+-----------------------------------------------------------------------+
```

Driver not loading in host server

What this looks like in dmsg:

```
[275604.216789] mlx5_core 0000:af:00.1: 63.008 Gb/s available PCIe bandwidth, limited by 8 GT/s x8 link at 0000:ae:00.0 (capable of 126.024 Gb/s with 16 GT/s x8 link)
[275624.187596] mlx5_core 0000:af:00.1: wait_fw_init:316:(pid 943): Waiting for FW initialization, timeout abort in 100s
[275644.152994] mlx5_core 0000:af:00.1: wait_fw_init:316:(pid 943): Waiting for FW initialization, timeout abort in 79s
[275664.118404] mlx5_core 0000:af:00.1: wait_fw_init:316:(pid 943): Waiting for FW initialization, timeout abort in 59s
[275684.083806] mlx5_core 0000:af:00.1: wait_fw_init:316:(pid 943): Waiting for FW initialization, timeout abort in 39s
[275704.049211] mlx5_core 0000:af:00.1: wait_fw_init:316:(pid 943): Waiting for FW initialization, timeout abort in 19s
[275723.954752] mlx5_core 0000:af:00.1: mlx5_function_setup:1237:(pid 943): Firmware over 120000 MS in pre-initializing state, aborting
[275723.968261] mlx5_core 0000:af:00.1: init_one:1813:(pid 943): mlx5_load_one failed with error code -16
[275723.978578] mlx5_core: probe of 0000:af:00.1 failed with error -16
```

The driver on the host server is dependent on the Arm side. If the driver on Arm is up, then the driver on the host server will also be up.

Please verify that:

- The driver is loaded in the BlueField (Arm)
- The Arm is booted into OS

- The Arm is not in UEFI Boot Menu
- The Arm is not hanged

Then:

1. Perform a graceful shutdown and a power cycle on the host server.
2. If the problem persists, reset nvconfig (`sudo mlxconfig -d /dev/mst/<device> -y reset`) and perform a BlueField system reboot.

   Note

   If your BlueField is VPI capable, please be aware that this configuration will reset the link type on the network ports to IB. To change the network port's link type to Ethernet, run:

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

   This configuration change requires performing a BlueField system reboot.

3. If this problem still persists, please make sure to install the latest bfb image and then restart the driver in host server. Please refer to "Installing Repo Package on Host Side" for more information.

No connectivity between network interfaces of source host to destination device

Verify that the bridge is configured properly on the Arm side.

The following is an example for default configuration:

```
$ sudo ovs-vsctl show
```

```
f6740bfb-0312-4cd8-88c0-a9680430924f
    Bridge ovsbr1
        Port pf0sf0
            Interface pf0sf0
        Port p0
            Interface p0
        Port pf0hpf
            Interface pf0hpf
        Port ovsbr1
            Interface ovsbr1
                type: internal
    Bridge ovsbr2
        Port p1
            Interface p1
        Port pf1sf0
            Interface pf1sf0
        Port pf1hpf
            Interface pf1hpf
        Port ovsbr2
            Interface ovsbr2
                type: internal
    ovs_version: "2.14.1"
```

If no bridge configuration exists, please refer to "Virtual Switch on BlueField".

Uplink in Arm down while uplink in host server up

Please check that the cables are connected properly into the network ports of BlueField and the peer device.

Performance Troubleshooting

Degradation in performance

Degradation in performance indicates that openvswitch may not be offloaded.

Verify offload state. Run:

```
# ovs-vsctl get Open_vSwitch . other_config:hw-offload
```

- If `hw-offload = true` – Fast Pass is configured (desired result)
- If `hw-offload = false` – Slow Pass is configured

If `hw-offload = false` :

- For RHEL/CentOS, run:

  ```
  # ovs-vsctl set Open_vSwitch . other_config:hw-offload=true;
  # systemctl restart openvswitch;
  # systemctl enable openvswitch;
  ```

- Ubuntu/Debian:

  ```
  # ovs-vsctl set Open_vSwitch . other_config:hw-offload=true;
  # /etc/init.d/openvswitch-switch restart
  ```

PCIe Troubleshooting and How-Tos

Insufficient power on the PCIe slot error

If the error "insufficient power on the PCIe slot" is printed in dmsg, please refer to the Specifications section of your hardware user guide and make sure that you are providing your DPU the correct amount of power.

To verify how much power is supported on your host's PCIe slots, run the command `lspci -vvv | grep PowerLimit`. For example:

```
# lspci -vvv | grep PowerLimit
                Slot #6, PowerLimit 75.000W; Interlock- NoCompl-
                Slot #1, PowerLimit 75.000W; Interlock- NoCompl-
                Slot #4, PowerLimit 75.000W; Interlock- NoCompl-
```

Note

Be aware that this command is not supported by all host vendors/types.

HowTo update PCIe device description

lspci may not present the full description for the NVIDIA PCIe devices connected to your host. For example:

```
# lspci | grep -i Mellanox
```

```
a3:00.0 Infiniband controller: Mellanox Technologies Device a2d6 (rev 01)
a3:00.1 Infiniband controller: Mellanox Technologies Device a2d6 (rev 01)
a3:00.2 DMA controller: Mellanox Technologies Device c2d3 (rev 01)
```

Please run the following command:

```
# update-pciids
```

Now you should be able to see the full description for those devices. For example:

```
# lspci | grep -i Mellanox
a3:00.0 Infiniband controller: Mellanox Technologies MT42822 BlueField-2 integrated ConnectX-6 Dx network controller (rev 01)
a3:00.1 Infiniband controller: Mellanox Technologies MT42822 BlueField-2 integrated ConnectX-6 Dx network controller (rev 01)
a3:00.2 DMA controller: Mellanox Technologies MT42822 BlueField-2 SoC Management Interface (rev 01)
```

HowTo handle two BlueField DPU devices in the same server

Please refer to section "Multi-board Management Example".

SR-IOV Troubleshooting

Unable to create VFs

1. Please make sure that SR-IOV is enabled in BIOS.
2. Verify SRIOV_EN is true and NUM_OF_VFS bigger than 1. Run:

```
# mlxconfig -d /dev/mst/mt41686_pciconf0 -e q |grep -i "SRIOV_EN\|num_of_vf"
Configurations:         Default       Current       Next Boot
*       NUM_OF_VFS      16            16            16
*       SRIOV_EN        True(1)       True(1)       True(1)
```

3. Verify that GRUB_CMDLINE_LINUX="iommu=pt intel_iommu=on pci=assign-busses".

No traffic between VF to external host

1. Please verify creation of representors for VFs inside the NVIDIA® BlueField® networking platform (DPU or SuperNIC). Run:

```
# /opt/mellanox/iproute2/sbin/rdma link |grep -i up
...
link mlx5_0/2 state ACTIVE physical_state LINK_UP netdev pf0vf0
...
```

2. Make sure the representors of the VFs are added to the bridge. Run:

```
# ovs-vsctl add-port <bridage_name> pf0vf0
```

3. Verify VF configuration. Run:

```
$ ovs-vsctl show
bb993992-7930-4dd2-bc14-73514854b024
    Bridge ovsbr1
        Port pf0vf0
            Interface pf0vf0
                type: internal
        Port pf0hpf
            Interface pf0hpf
        Port pf0sf0
            Interface pf0sf0
        Port p0
            Interface p0
    Bridge ovsbr2
        Port ovsbr2
            Interface ovsbr2
                type: internal
        Port pf1sf0
            Interface pf1sf0
        Port p1
            Interface p1
        Port pf1hpf
            Interface pf1hpf
    ovs_version: "2.14.1"
```

eSwitch Troubleshooting

Unable to configure legacy mode

To set devlink to "Legacy" mode in NVIDIA® BlueField® networking platform (DPU or SuperNIC), run:

```
# devlink dev eswitch set pci/0000:03:00.0 mode legacy
# devlink dev eswitch set pci/0000:03:00.1 mode legacy
```

Please verify that:

- No virtual functions are open. To verify if VFs are configured, run:

  ```
  # /opt/mellanox/iproute2/sbin/rdma link | grep -i up
  link mlx5_0/2 state ACTIVE physical_state LINK_UP netdev pf0vf0
  link mlx5_1/2 state ACTIVE physical_state LINK_UP netdev pf1vf0
  ```

  If any VFs are configured, destroy them by running:

  ```
  # echo 0 > /sys/class/infiniband/mlx5_0/device/mlx5_num_vfs
  # echo 0 > /sys/class/infiniband/mlx5_1/device/mlx5_num_vfs
  ```

- If any SFs are configured, delete them by running:

  ```
  /sbin/mlnx-sf -a delete --sfindex <SF Index>
  ```

  Note

You may retrieve the <SF Index> of the currently installed SFs by running:

```
# mlnx-sf -a show

SF Index: pci/0000:03:00.0/229408
  Parent PCI dev: 0000:03:00.0
  Representor netdev: en3f0pf0sf0
  Function HWADDR: 02:61:f6:21:32:8c
  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: 02:30:13:6a:2d:2c
  Auxiliary device: mlx5_core.sf.3
    netdev: enp3s0f1s0
    RDMA dev: mlx5_3
```

Pay attention to the SF Index values. For example:

```
/sbin/mlnx-sf -a delete --sfindex pci/0000:03:00.0/229408
/sbin/mlnx-sf -a delete --sfindex pci/0000:03:00.1/294944
```

If the error "Error: mlx5_core: Can't change mode when flows are configured" is encountered while trying to configure legacy mode, please make sure that

1. Any configured SFs are deleted (see above for commands).
2. Shut down the links of all interfaces, delete any ip xfrm rules, delete any configured OVS flows, and stop openvswitch service. Run:

```
ip link set dev p0 down
ip link set dev p1 down
ip link set dev pf0hpf down
ip link set dev pf1hpf down
ip link set dev vxlan_sys_4789 down

ip x s f ;
ip x p f ;

tc filter del dev p0 ingress
tc filter del dev p1 ingress
tc qdisc show dev p0
tc qdisc show dev p1
tc qdisc del dev p0 ingress
tc qdisc del dev p1 ingress
tc qdisc show dev p0
tc qdisc show dev p1

systemctl stop openvswitch-switch
```

Arm appears as two interfaces

What this looks like:

```
# sudo /opt/mellanox/iproute2/sbin/rdma link
link mlx5_0/1 state ACTIVE physical_state LINK_UP netdev p0
link mlx5_1/1 state ACTIVE physical_state LINK_UP netdev p1
```

- Check if you are working in legacy mode.

  ```
  # devlink dev eswitch show pci/0000:03:00.<0|1>
  ```

  If the following line is printed, this means that you are working in legacy mode:

  ```
  pci/0000:03:00.<0|1>: mode legacy inline-mode none encap enable
  ```

  Please configure the DPU to work in switchdev mode. Run:

```
devlink dev eswitch set pci/0000:03:00.<0|1> mode switchdev
```

- Check if you are working in separated mode:

```
# mlxconfig -d /dev/mst/mt41686_pciconf0 q | grep -i cpu
* INTERNAL_CPU_MODEL SEPERATED_HOST(0)
```

  Please configure the DPU to work in embedded mode. Run:

```
devlink dev eswitch set pci/0000:03:00.<0|1> mode switchdev
```

Isolated Mode Troubleshooting and How-Tos

Unable to burn FW from host server

Please verify that you are not in running in isolated mode. Run:

```
$ sudo mlxprivhost -d /dev/mst/mt41686_pciconf0 q
Current device configurations:
------------------------------
level                    : PRIVILEGED
...
```

General Troubleshooting

Server unable to find the BlueField

- Ensure that the NVIDIA BlueField networking platform (DPU or SuperNIC) is placed correctly
- Make sure the BlueField slot and the BlueField are compatible
- Install BlueField in a different PCI Express slot
- Use the drivers that came with BlueField or download the latest
- Make sure your motherboard has the latest BIOS
- Perform a graceful shutdown then power cycle the server

BlueField no longer works

- Reseat BlueField in its slot or a different slot, if necessary
- Try using another cable
- Reinstall the drivers for the network driver files may be damaged or deleted
- Perform a graceful shutdown then power cycle the server

BlueField stopped working after installing another BFB

- Try removing and reinstalling all BlueFields
- Check that cables are connected properly
- Make sure your motherboard has the latest BIOS

Link indicator light is off

- Try another port on the switch
- Make sure the cable is securely attached
- Check you are using the proper cables that do not exceed the recommended lengths
- Verify that your switch and BlueField port are compatible

Link light is on but no communication is established

- Check that the latest driver is loaded
- Check that both BlueField and its link are set to the same speed and duplex settings

Installation Troubleshooting and How-Tos

BlueField target is stuck inside UEFI menu

Upgrade to the latest stable boot partition images, see "How to upgrade the boot partition (ATF & UEFI) without re-installation".

BFB does not recognize the BlueField board type

If the .bfb file cannot recognize the BlueField board type, it reverts to low core operation. The following message will be printed on your screen:

```
***System type can't be determined***
***Booting as a minimal system***
```

Please contact NVIDIA Support if this occurs.

Unable to load BL2, BL2R, or PSC image

The following errors appear in console if images are corrupted or not signed properly:

Device	Error
BlueField	ERROR: Failed to load BL2 firmware
BlueField-2	ERROR: Failed to load BL2R firmware
BlueField-3	Failed to load PSC-BL1 or PSC VERIFY_BCT timeout

CentOS fails into "dracut" mode during installation

This is most likely configuration related.

- If installing through the RShim interface, check whether /var/pxe/centos7 is mounted or not. If not, either manually mount it or re-run the setup.sh script.
- Check the Linux boot message to see whether eMMC is found or not. If not, the BlueField driver patch is missing. For local installation via RShim, run the setup.sh script with the absolute path and check if there are any errors. For a corporate PXE server, make sure the BlueField and ConnectX driver disk are patched into the initrd image.

How to find the software versions of the running system

Run the following:

```
/opt/mellanox/scripts/bfvcheck:
root@bluefield:/usr/bin/bfvcheck# ./bfvcheck
Beginning version check...
-RECOMMENDED VERSIONS-
ATF: v1.5(release):BL2.0-1-gf9f7cdd
UEFI: 2.0-6004a6b
FW: 18.25.1010
-INSTALLED VERSIONS-
ATF: v1.5(release):BL2.0-1-gf9f7cdd
UEFI: 2.0-6004a6b
FW: 18.25.1010
Version checked
```

Also, the version information is printed to the console.

For ATF, a version string is printed as the system boots.

```
"NOTICE:  BL2: v1.3(release):v1.3-554-ga622cde"
```

For UEFI, a version string is printed as the system boots.

```
"UEFI firmware (version 0.99-18d57e3 built at 00:55:30 on Apr 13 2018)"
```

For Yocto, run:

```
$ cat /etc/bluefield_version
2.0.0.10817
```

How to upgrade the host RShim driver

See the readme at `<BF_INST_DIR>/src/drivers/rshim/README`.

How to upgrade the boot partition (ATF & UEFI) without re-installation

1. Boot the target through the RShim interface from a host machine:

   ```
   $ cat <BF_INST_DIR>/sample/install.bfb > /dev/rshim<N>/boot
   ```

2. Log into the BlueField target:

   ```
   $ /opt/mlnx/scripts/bfrec
   ```

How to upgrade ConnectX firmware from Arm side

The `mst`, `mlxburn`, and `flint` tools can be used to update firmware.

For Ubuntu, CentOS and Debian, run the following command from the Arm side:

```
sudo /opt/mellanox/mlnx-fw-updater/mlnx_fw_updater.pl
```

How to configure ConnectX firmware

Configuring ConnectX firmware can be done using the mlxconfig tool.

It is possible to configure privileges of both the internal (Arm) and the external host (for DPUs) from a privileged host. According to the configured privilege, a host may or may not perform certain operations related to the NIC (e.g. determine if a certain host is allowed to read port counters).

For more information and examples please refer to the MFT User Manual which can be found at the following link.

How to use the UEFI boot menu

Press the "Esc" key when prompted after booting (before the countdown timer runs out) to enter the UEFI boot menu and use the arrows to select the menu option.

It could take 1-2 minutes to enter the Boot Manager depending on how many devices are installed or whether the EXPROM is programmed or not.

Once in the boot manager:

- "EFI Network xxx" entries with device path "PciRoot..." are ConnectX interface
- "EFI Network xxx" entries with device path "MAC(..." are for the RShim interface and the BlueField OOB Ethernet interface

Select the interface and press ENTER will start PXE boot.

The following are several useful commands under UEFI shell:

```
Shell> ls FS0:                          # display file
Shell> ls FS0:\EFI                      # display file
Shell> cls                              # clear screen
Shell> ifconfig -l                      # show interfaces
Shell> ifconfig -s eth0 dhcp            # request DHCP
Shell> ifconfig -l eth0                         # show one interface
Shell> tftp 192.168.100.1 grub.cfg FS0:\grub.cfg    # tftp download a file
Shell> bcfg boot dump                           # dump boot variables
Shell> bcfg boot add 0 FS0:\EFI\centos\shim.efi "CentOS" # create an entry
```

How to Use the Kernel Debugger (KGDB)

The default Yocto kernel has `CONFIG_KGDB` and `CONFIG_KGDB_SERIAL_CONSOLE` enabled. This allows the Linux kernel on BlueField to be debugged over the serial port. A single serial port cannot be used both as a console and by KGDB at the same time. It is recommended to use the RShim for console access (`/dev/rshim0/console`) and the UART port (`/dev/ttyAMA0 or /dev/ttyAMA1`) for KGDB. Kernel GDB over console (KGDBOC) does not work over the RShim console. If the RShim console is not available, there are open-source packages such as KGDB demux and agent-proxy which allow a single serial port to be shared.

There are two ways to configure KGDBOC. If the OS is already booted, then write the name of the serial device to the KGDBOC module parameter. For example:

```
$ echo ttyAMA1 > /sys/module/kgdboc/parameters/kgdboc
```

To attach GDB to the kernel, it must be stopped first. One way to do that is to send a "g" to `/proc/sysrq-trigger`.

```
$ echo g > /proc/sysrq-trigger
```

To debug incidents that occur at boot time, kernel boot parameters must be configured. Add "`kgdboc=ttyAMA1,115200 kgdwait`" to the boot arguments to use UART1 for debugging and force it to wait for GDB to attach before booting.

Once the KGDBOC module is configured and the kernel stopped, run the Arm64 GDB on the host machine connected to the serial port, then set the remote target to the serial device on the host side.

```
<BF_INST_DIR>/sdk/sysroots/x86_64-pokysdk-linux/usr/bin/aarch64-poky-linux/aarch64-poky-linux-gdb
<BF_INST_DIR>/sample/vmlinux
```

```
(gdb) target remote /dev/ttyUSB3
Remote debugging using /dev/ttyUSB3
arch_kgdb_breakpoint () at /labhome/dwoods/src/bf/linux/arch/arm64/include/asm/kgdb.h:32
32          asm ("brk %0" : : "I" (KGDB_COMPILED_DBG_BRK_IMM));
(gdb)
```

`<BF_INST_DIR>` is the directory where the BlueField software is installed. It is assumed that the SDK has been unpacked in the same directory.

How to enable/disable SMMU

SMMU could affect performance for certain applications. By default, it is enabled on BlueField-3 and disabled on BlueField-2, and can be configured in different ways.

- Enable/disable SMMU in the UEFI System Configuration
- Set it in `bf.cfg` and push it together with the `install.bfb` (see section "Installing Popular Linux Distributions on BlueField")
- In BlueField Linux, create a file with one line with `SYS_ENABLE_SMMU=TRUE`, then run `bfcfg`.

The configuration change will take effect after reboot. The configuration value is stored in a persistent UEFI variable. It is not modified by OS installation.

See section "UEFI System Configuration" for information on how to access the UEFI System Configuration menu.

How to change the default console of the install image

On UART0:

```
$ echo "console=ttyAMA0 earlycon=pl011,0x01000000 initrd=initramfs" > bootarg
$ <BF_INST_DIR>/bin/mlx-mkbfb --boot-args bootarg \
      <BF_INST_DIR>/sample/ install.bfb
```

On UART1:

```
$ echo "console=ttyAMA1 earlycon=pl011,0x01000000 initrd=initramfs" > bootarg
$ <BF_INST_DIR>/bin/mlx-mkbfb --boot-args bootarg \
      <BF_INST_DIR>/sample/install.bfb
```

On RShim:

```
$ echo "console=hvc0 initrd=initramfs" > bootarg
$ <BF_INST_DIR>/bin/mlx-mkbfb --boot-args bootarg \
      <BF_INST_DIR>/sample/install.bfb
```

How to change the default network configuration during BFB installation

On Ubuntu OS, the default network configuration for `tmfifo_net0` and `oob_net0` interfaces is set by the cloud-init service upon first boot after BFB installation.

The default content of `/var/lib/cloud/seed/nocloud-net/network-config` as follows:

```
# cat /var/lib/cloud/seed/nocloud-net/network-config
version: 2
renderer: NetworkManager
ethernets:
  tmfifo_net0:
    dhcp4: false
    addresses:
      - 192.168.100.2/30
    nameservers:
      addresses: [ 192.168.100.1 ]
    routes:
    - to: 0.0.0.0/0
      via: 192.168.100.1
      metric: 1025
  oob_net0:
    dhcp4: true
```

This content can be modified during BFB installation using bf.cfg. For example:

```
# cat bf.cfg
bfb_modify_os()
{
    sed -i -e '/oob_net0/,+1d' /mnt/var/lib/cloud/seed/nocloud-net/network-config
cat >> /mnt/var/lib/cloud/seed/nocloud-net/network-config << EOF
  oob_net0:
    dhcp4: false
    addresses:
      - 10.0.0.1/24
EOF
}

# bfb-install  -c bf.cfg -r rshim0 -b <BFB>
```

Note

Using the same technique, any configuration file on the BlueField DPU side can be updated during the BFB installation process.

Sanitizing DPU eMMC and SSD Storage

During the BFB installation process, NVIDIA® BlueField® networking platform (DPU or SuperNIC) storage can be securely sanitized either using the shred or the mmc and nvme utilities in the bf.cfg configuration file as illustrated in the following subsections.

Note

By default, only the installation target storage is formatted using the Linux mkfs utility.

Using shred Utility

```
# cat bf.cfg
SANITIZE_DONE=${SANITIZE_DONE:-0}
export SANITIZE_DONE
if [ $SANITIZE_DONE -eq 0 ]; then
        sleep 3m
        /sbin/modprobe nvme

        if [ -e /dev/mmcblk0 ]; then
                echo Sanitizing /dev/mmcblk0 | tee /dev/kmsg
                echo Sanitizing /dev/mmcblk0 > /tmp/sanitize.emmc.log
                mmc sanitize /dev/mmcblk0 >>  /tmp/sanitize.emmc.log 2>&1
        fi
        if [ -e /dev/nvme0n1 ]; then
                echo Sanitizing /dev/nvme0n1 | tee /dev/kmsg
                echo Sanitizing /dev/nvme0n1 > /tmp/sanitize.ssd.log
                nvme sanitize /dev/nvme0n1 -a 2 >> /tmp/sanitize.ssd.log 2>&1
                nvme sanitize-log /dev/nvme0n1 >> /tmp/sanitize.ssd.log 2>&1
        fi
        SANITIZE_DONE=1
        echo ===================== sanitize.log ===================== | tee /dev/kmsg
        cat /tmp/sanitize.*.log | tee /dev/kmsg
        sync
fi
bfb_modify_os()
{
        echo ===================== bfb_modify_os ===================== | tee /dev/kmsg
        if ( /bin/ls -1 /tmp/sanitize.*.log > /dev/null 2>&1 ); then
                cat /tmp/sanitize.*.log > /mnt/root/sanitize.log
        fi
}
```

Using mmc and nvme Utilities

```
# cat bf.cfg
SANITIZE_DONE=${SANITIZE_DONE:-0}
export SANITIZE_DONE
```

```
if [ $SANITIZE_DONE -eq 0 ]; then
        sleep 3m
        /sbin/modprobe nvme

        if [ -e /dev/mmcblk0 ]; then
                echo Sanitizing /dev/mmcblk0 | tee /dev/kmsg
                echo Sanitizing /dev/mmcblk0 > /tmp/sanitize.emmc.log
                mmc sanitize /dev/mmcblk0 >>  /tmp/sanitize.emmc.log 2>&1
        fi
        if [ -e /dev/nvme0n1 ]; then
                echo Sanitizing /dev/nvme0n1 | tee /dev/kmsg
                echo Sanitizing /dev/nvme0n1 > /tmp/sanitize.ssd.log
                nvme sanitize /dev/nvme0n1 -a 2 >> /tmp/sanitize.ssd.log 2>&1
                nvme sanitize-log /dev/nvme0n1 >> /tmp/sanitize.ssd.log 2>&1
        fi
        SANITIZE_DONE=1
        echo ===================== sanitize.log ===================== | tee /dev/kmsg
        cat /tmp/sanitize.*.log | tee /dev/kmsg
        sync
fi
bfb_modify_os()
{
        echo ===================== bfb_modify_os ===================== | tee /dev/kmsg
        if ( /bin/ls -1 /tmp/sanitize.*.log > /dev/null 2>&1 ); then
                cat /tmp/sanitize.*.log > /mnt/root/sanitize.log
        fi
}
```

Ubuntu Kernel Debug

Installing Kernel Debug Symbols

To install kernel debug symbols, run the following:

```
# sudo add-apt-repository ppa:canonical-kernel-bluefield/release
# echo "deb https://ppa.launchpadcontent.net/canonical-kernel-bluefield/release/ubuntu/ jammy
main/debug" | sudo tee -a /etc/apt/sources.list.d/canonical-kernel-bluefield-ubuntu-release-jammy.list
# sudo apt update
# sudoapt install linux-image-$(uname -r)-dbgsym
E.g.:
# sudo apt-cache policy linux-image-unsigned-5.15.0-1043-bluefield-dbgsym
# sudo apt install linux-image-unsigned-5.15.0-1043-bluefield-dbgsym
```