Modes of Operation

Table of contents

The NVIDIA® BlueField® networking platform (DPU or SuperNIC) has several modes of operation:

- DPU mode, or embedded function (ECPF) ownership, where the embedded Arm system controls the NIC resources and data path
- Zero-trust mode which is an extension of the ECPF ownership with additional restrictions on the host side
- NIC mode where BlueField behaves exactly like an adapter card from the perspective of the external host

> **Note**
>
> The default mode of operation for the BlueField DPU is DPU mode
>
> The default mode of operation for the BlueField SuperNIC is NIC mode

DPU Mode

This mode, known also as embedded CPU function ownership (ECPF) mode, is the default mode for the BlueField DPU.

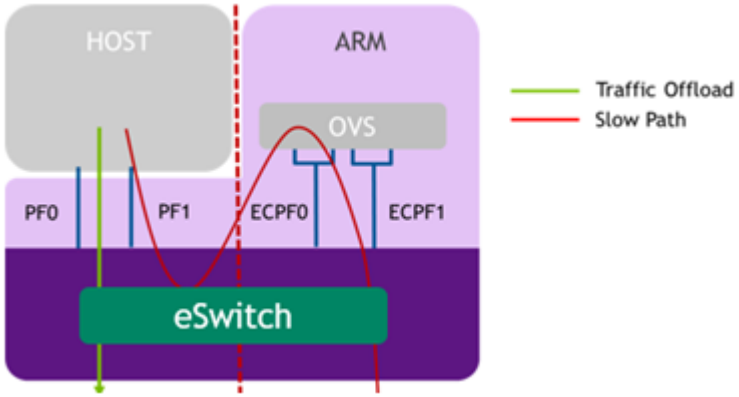
In DPU mode, the NIC resources and functionality are owned and controlled by the embedded Arm subsystem. All network communication to the host flows through a virtual switch control plane hosted on the Arm cores, and only then proceeds to the host. While working in this mode, the BlueField is the trusted function managed by the data center and host administrator—to load network drivers, reset an interface, bring an interface up and down, update the firmware, and change the mode of operation on BlueField.

A network function is still exposed to the host, but it has limited privileges. In particular:

1. The driver on the host side can only be loaded after the driver on the BlueField has loaded and completed NIC configuration.

2. All ICM (Interface Configuration Memory) is allocated by the ECPF and resides in the BlueField's memory.

3. The ECPF controls and configures the NIC embedded switch which means that traffic to and from the host (BlueField) interface always lands on the Arm side.



When the server and BlueField are initiated, the networking to the host is blocked until the virtual switch on the BlueField is loaded. Once it is loaded, traffic to the host is allowed by default.

There are two ways to pass traffic to the host interface: Either using representors to forward traffic to the host (every packet to/from the host would be handled also by the network interface on the embedded Arm side) or push rules to the embedded switch which allows and offloads this traffic.

In DPU mode, OpenSM must be run from the BlueField side (not the host side). Also, management tools (e.g., sminfo, ibdev2netdev, ibnetdiscover) can only be run from the BlueField side (not from the host side).

Zero-trust Mode

Zero-trust mode is a specialization of DPU mode which implements an additional layer of security where the host system administrator is prevented from accessing BlueField from the host. Once zero-trust mode is enabled, the data center administrator should control BlueField entirely through the Arm cores and/or BMC connection instead of through the host.

For security and isolation purposes, it is possible to restrict the host from performing operations that can compromise the BlueField. The following operations can be restricted individually when changing the BlueField host to zero-trust mode:

- Port ownership – the host cannot assign itself as port owner
- Hardware counters – t he host does not have access to hardware counters
- Tracer functionality is blocked
- RShim interface is blocked
- Firmware flash is restricted

Enabling Zero-trust Mode

To enable host restriction:

1. Start the MST service.
2. Set zero-trust mode. From the Arm side, run:

```
$ sudo mlxprivhost -d /dev/mst/<device> r --disable_rshim --disable_tracer --disable_counter_rd --disable_port_owner
```

   - If no --disable_* flags are used, users must perform BlueField system reboot .
   - If any --disable_* flags are used, users must perform BlueField system-level reset.

Disabling Zero-trust Mode

To disable host restriction:

1. Set the mode to privileged. Run:

```
$ sudo mlxprivhost -d /dev/mst/<device> p
```

2. Applying configuration:

- I f host restriction had not been applied using any --disable_* flags, users must perform BlueField system reboot .
- If host restriction had been applied using any --disable_* flags, users must perform BlueField system-level reset.

NIC Mode

In this mode, BlueField behaves exactly like an adapter card from the perspective of the external host.

> **Note**
>
> The following instructions presume BlueField to be operating in DPU mode. If BlueField is operating in zero-trust mode, please return to DPU mode before proceeding.

> **Note**
>
> The following notes are relevant for updating the BFB bundle in NIC mode:
>
> - During BFB Bundle installation, Linux is expected to boot to upgrade NIC firmware and BMC software
> - During the BFB Bundle installation, it is expected for the mlx5 driver to error messages on the x86 host. These prints may be ignored as they are resolved by a mandatory, post-installation power cycle.
> - It is mandatory to power cycle the host after the installation is complete for the changes to take effect

- As Linux is booting during BFB Bundle installation, it is expected for the mlx5 core driver to timeout on the BlueField Arm

NIC Mode for BlueField-3

Note

When BlueField-3 is configured to operate in NIC mode, Arm OS will not boot.

NIC mode for BlueField-3 saves power, improves device performance, and improves the host memory footprint.

Configuring NIC Mode on BlueField-3 from Linux

Enabling NIC Mode on BlueField-3 from Linux

Before moving to NIC mode, make sure you are operating in DPU mode by running:

```
host/bf> sudo mlxconfig -d /dev/mst/mt41692_pciconf0 -e q
```

The output should have INTERNAL_CPU_MODEL= EMBBEDDED_CPU(1) and EXP_ROM_UEFI_ARM_ENABLE = True (1) (default).

To enable NIC mode from DPU mode:

1. Run the following on the host or Arm:

   ```
   host/bf> sudo mlxconfig -d /dev/mst/mt41692_pciconf0 s INTERNAL_CPU_OFFLOAD_ENGINE=1
   ```

2. Perform BlueField system-level reset for the mlxconfig settings to take effect.

Disabling NIC Mode on BlueField-3 from Linux

To return to DPU mode from NIC mode:

1. Run the following on the host:

```
host> sudo mlxconfig -d /dev/mst/mt41692_pciconf0 s INTERNAL_CPU_OFFLOAD_ENGINE=0
```

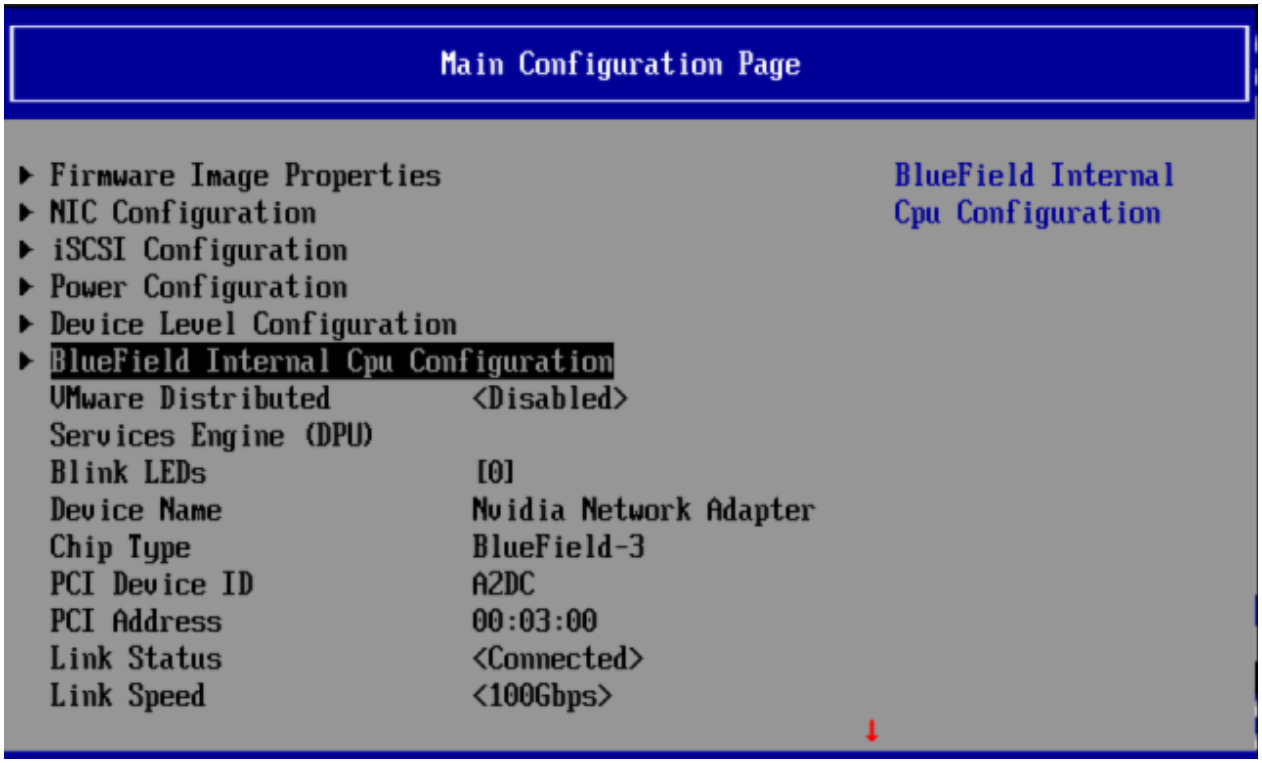
2. Perform BlueField system-level reset for the mlxconfig settings to take effect.

Configuring NIC Mode on BlueField-3 from Host BIOS HII UEFI Menu

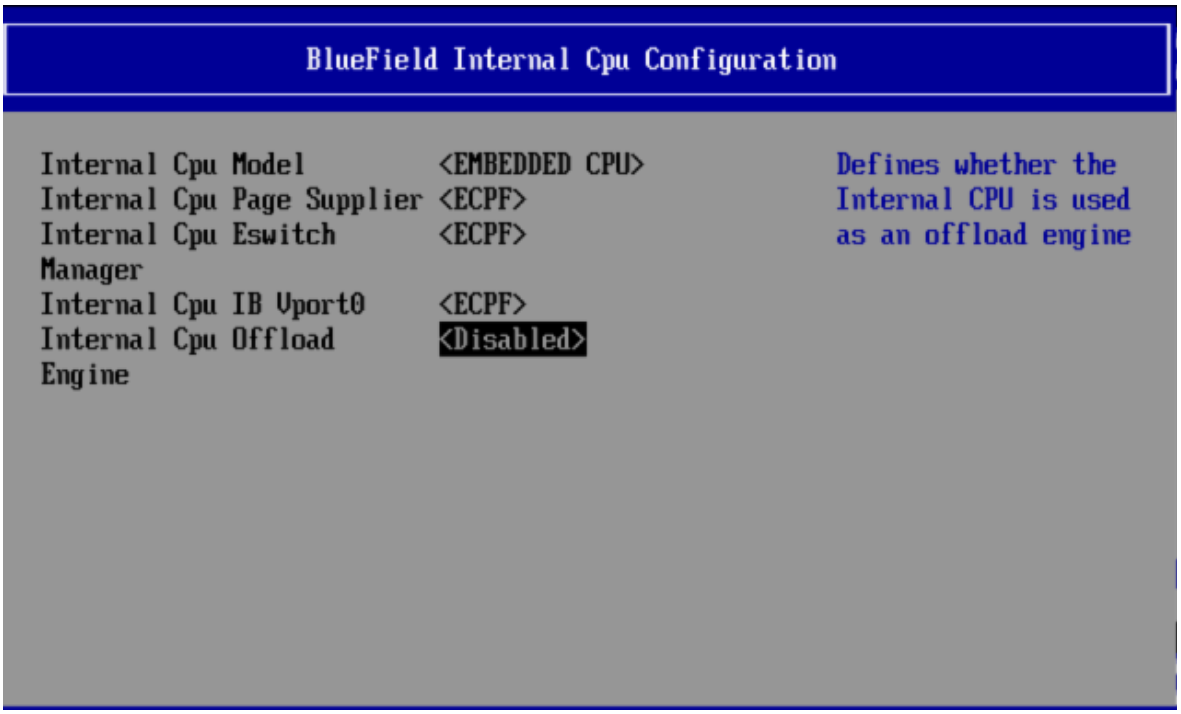
> **Info**
>
> The screenshots in this section are examples only and may vary depending on the vendor of your specific host.

1. Select the network device that presents the uplink (i.e., select the device with the uplink MAC address).

2. Select "BlueField Internal Cpu Configuration".



- To enable NIC mode, set "Internal Cpu Offload Engine" to "Disabled".
- To switch back to DPU mode, set "Internal Cpu Offload Engine" to "Enabled".



Configuring NIC Mode on BlueField-3 from Arm UEFI

1. Access the Arm UEFI menu by pressing the Esc button twice.
2. Select "Device Manager".
3. Select "System Configuration".
4. Select "BlueField Modes".
5. Set the "NIC Mode" field to NicMode to enable NIC mode.

> **Info**
>
> Configuring Unavailable is inapplicable.

6. Exit "BlueField Modes" and "System Configuration" and make sure to save the settings. Exit the UEFI setup using the 'reset' option. The configuration is not yet applied and BlueField is expected to boot regularly, still in DPU mode.
7. perform BlueField system-level reset to change to NIC mode.

Configuring NIC Mode on BlueField-3 Using Redfish

Run the following from the BlueField BMC:

1. Get the current BIOS attributes:

```
sudo curl -k -u root:'<password>' -H 'content-type: application/json' -X GET
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Bios/
```

2. Change BlueField mode from DpuMode to NicMode:

```
curl -k -u root:'<password>' -H 'content-type: application/json' -d '{ "Attributes": { "NicMode":
"NicMode" } }' -X PATCH https://<bmc_ip>/redfish/v1/Systems/Bluefield/Bios/Settings
```

> **Info**
>
> To revert back to DPU mode, run:
>
> ```
> curl -k -u root:'<password>' -H 'content-type: application/json' -d '{
> "Attributes": { "NicMode": "DpuMode" } }' -X PATCH
> https://<bmc_ip>/redfish/v1/Systems/Bluefield/Bios/Settings
> ```

3. Verify that the BMC has registered the new settings:

```
curl -k -u root:'<password>' -H 'content-type: application/json' -X GET
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Bios/Settings
```

4. Issue a software reset then power cycle the host for the change to take effect.

5. Verify the mode is changed:

```
curl -k -u root:'<password>' -H 'content-type: application/json' -X GET
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Oem/Nvidia
```

> **Note**
>
> To retrieve the mode via BIOS attributes, another BlueField software reset is required before running the command:
>
> ```
> curl -k -u root:'<password>' -H 'content-type: application/json' -X GET https://<bmc_ip>/redfish/v1/Systems/Bluefield/Bios
> ```

Updating Firmware Components in BlueField-3 NIC Mode

Once in NIC mode, updating ATF and UFEI can be done using the standard *.bfb image:

```
# bfb-install --bfb <BlueField-BSP>.bfb --rshim rshim0
```

NIC Mode for BlueField-2

In this mode, the ECPFs on the Arm side are not functional but the user is still able to access the Arm system and update mlxconfig options.

> **Note**
>
> When NIC mode is enabled, the drivers and services on the Arm are no longer functional.

Configuring NIC Mode on BlueField-2 from Linux

Enabling NIC Mode on BlueField-2 from Linux

To enable NIC mode from DPU mode:

1. Run the following from the x86 host side:

```
$ mst start
$ mlxconfig -d /dev/mst/<device> s \
INTERNAL_CPU_PAGE_SUPPLIER=1 \
INTERNAL_CPU_ESWITCH_MANAGER=1 \
INTERNAL_CPU_IB_VPORT0=1 \
INTERNAL_CPU_OFFLOAD_ENGINE=1
```

> **Note**
>
> To restrict RShim PF (optional), make sure to configure INTERNAL_CPU_RSHIM=1 as part of the mlxconfig command.

2. Perform BlueField system-level reset t o load the new configuration .

> **Info**
>
> Refer to the troubleshooting section of the guide for a step-by-step procedure.

> **Note**
>
> Multi-host is not supported when BlueField is operating in NIC mode.

> **Note**
>
> To obtain firmware BINs for BlueField-2 devices, please refer to the BlueField-2 firmware download page.

Disabling NIC Mode on BlueField-2 from Linux

To change from NIC mode back to DPU mode:

1. Install and start the RShim driver on the host.
2. Disable NIC mode. Run:

```
$ mst start
$ mlxconfig -d /dev/mst/<device> s \
INTERNAL_CPU_PAGE_SUPPLIER=0 \
INTERNAL_CPU_ESWITCH_MANAGER=0 \
INTERNAL_CPU_IB_VPORT0=0 \
INTERNAL_CPU_OFFLOAD_ENGINE=0
```

> **Note**
>
> If INTERNAL_CPU_RSHIM=1, then make sure to configure INTERNAL_CPU_RSHIM=0 as part of the mlxconfig command.

3. Perform a BlueField system reboot for the mlxconfig settings to take effect.

Configuring NIC Mode on BlueField-2 from Arm UEFI

Follow the same instructions in section "Configuring NIC Mode on BlueField-3 from Arm UEFI".

Configuring NIC Mode on BlueField-2 Using Redfish

Follow the same instructions in section "Configuring NIC Mode on BlueField-3 Using Redfish".

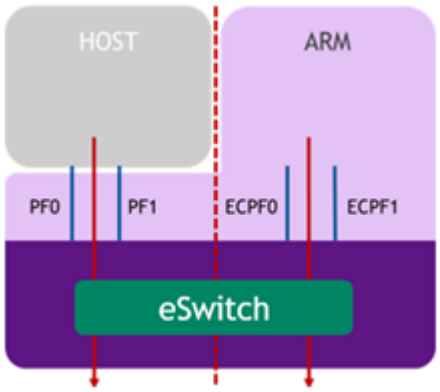
Separated Host Mode (Obsolete)

> ⚠ **Warning**
>
> This BlueField mode of operation is obsolete. Please do not use it!

In separated host mode, a network function is assigned to both the Arm cores and the host cores. The ports/functions are symmetric in the sense that traffic is sent to both physical functions simultaneously. Each one of those functions has its own MAC address, which allows one to communicate with the other, and can send and receive Ethernet and RDMA over Converged Ethernet (RoCE) traffic. There is an equal bandwidth share between the two functions.

There is no dependency between the two functions. They can operate simultaneously or separately. The host can communicate with the embedded function as two separate hosts, each with its own MAC and IP addresses (configured as a standard interface).

In separated host mode, the host administrator is a trusted actor who can perform all configuration and management actions related to either network function.



This mode enables the same operational model of a SmartNIC (that does not have a separated control plane). In this case, the Arm control plane can be used for different functions but does not have any control on the host steering functions.

The limitations of this mode are as follows:

- Switchdev (virtual switch offload) mode is not supported on either of the functions
- SR-IOV is only supported on the host side

To configure separated host mode from DPU mode:

1. Enable separated host mode. Run:

```
$ mst start
$ mlxconfig -d /dev/mst/<device> s INTERNAL_CPU_MODEL=0
```

2. Power cycle.

3. Verify configuration. Run:

```
$ mst start
$ mlxconfig -d /dev/mst/<device> q | grep -i model
```

4. Remove OVS bridges configuration from the Arm-side. Run:

```
$ ovs-vsctl list-br | xargs -r -l ovs-vsctl del-br
```