



NVIDIA BlueField Management and Initial Provisioning

Table of contents

Introduction	6
Prerequisites for Initial BlueField Deployment	11
First-time Installation Procedure	13
DPU Mode Installation	13
NIC Mode Installation	58
Day 2 Management Operations	62
Appendixes	71
Appendix - BMC and eROT Upgrade Process for BlueField-2	71
Appendix - BlueField Management in DPU Mode	73
Appendix - BlueField Management in NIC Mode	81
Appendix - BlueField DHCP Discover	85
Appendix - Using Boot Order Schema to Set Boot Order	89
Appendix - Relating BlueField to Host and Port	91

About This Document

This document defines the NVIDIA-recommended method to manage NVIDIA® BlueField®-2 and NVIDIA® BlueField®-3 networking platforms (DPUs or SuperNICs), reviews BlueField management interfaces, protocols, and capabilities (hardware, firmware, etc.), and explains how to use them to manage the BlueField.

This document also defines the NVIDIA-recommended method for initial provisioning and deploying of BlueField, along with a detailed explanation of the required steps and commands.

Target Audience

This document describes day 1 and 2 operations, and is intended for product managers, CSOs, CTOs, and data center administrators.

Scope

The applicable use case for this documents includes 1-2 BlueField DPUs in North-West connectivity operating in DPU mode and 8 BlueField SuperNICs in East-West connectivity operating in NIC mode.

Technical Support

Customers who purchased NVIDIA products directly from NVIDIA are invited to contact us through the following methods:

- E-mail: enterprisesupport@nvidia.com
- Enterprise Support page: <https://www.nvidia.com/en-us/support/enterprise>

Customers who purchased NVIDIA M-1 Global Support Services, please see your contract for details regarding technical support.

Customers who purchased NVIDIA products through an NVIDIA-approved reseller should first seek assistance through their reseller.

Document Conventions

This document uses angled brackets notation ('<>') to indicate variable parameters that depend on the user's specific setup or configuration which should be replaced accordingly.

The following are variables which make frequent appearances in this document:

Variable	Description
<BF-BMC-IP>	IP of the BlueField's BMC
<username>	Specific username
<password>	Relevant password (whether of a specific user or otherwise)
<System ID>	Used in Redfish requests, to obtain the value of this variable in your specific setup, run the following command: <pre>curl -vk -X GET -u root:<password> https://<BF-BMC-IP>/redfish/v1/Systems</pre> <p>Output example:</p> <pre>* ALPN, offering h2 * ALPN, offering http/1.1 * TLSv1.0 (OUT), TLS header, Certificate Status (22): ... { "@odata.id": "/redfish/v1/Systems", "@odata.type": "#ComputerSystemCollection.ComputerSystemCollection", "Members": [{ "@odata.id": "/redfish/v1/Systems/Bluefield" }], }</pre>

Variable	Description
	<pre>"Members@odata.count": 1, "Name": "Computer System Collection"</pre> <p>i Info SystemID is provided under Members: "@odata.id": "/redfish/v1/Systems/Bluefield".</p>

Glossary

Term	Description
BFB	BlueField bootstream
BM C	Board management controller
BSP	BlueField support package
DO CA	BlueField SDK
eM MC	Embedded multi-media card
eR OT	Extended enhanced root of trust <ul style="list-style-type: none"> • CEC for BlueField-2 • Glacier for BlueField-3
FW	Firmware
Host	When referring to "the host" this documentation is referring to the server host . When referring to the Arm based host, the documentation will specifically call out "Arm host".

Term	Description
	<ul style="list-style-type: none"> • Server host OS refers to the host server OS (Linux or Windows) • Arm host refers to the aarch64 Linux OS which is running on the BlueField Arm cores
HW	Hardware
NIC	Network interface card
OOB	Out-of-band
OS	Operating system
PCIe	PCI Express; Peripheral Component Interconnect Express
PF	Physical function
RMC	Remote management controller
UEFI	Unified extensible firmware interface
VF	Virtual function
VM	Virtual machine

Introduction

BlueField DPU

The NVIDIA® BlueField® DPU is a data center infrastructure on a chip that combines a high-speed networking interface with powerful, software-programmable Arm cores, enabling breakthrough networking, storage, and security performance. The BlueField DPU offloads, accelerates, and isolates a broad range of software-defined infrastructure services which traditionally ran on the host's CPU, overcoming performance and scalability bottlenecks, and eliminating security threats in modern data centers.

BlueField DPUs transform traditional computing environments to secure and accelerated data centers, allowing organizations to efficiently run data-driven, cloud-native applications alongside legacy applications. By decoupling the data center infrastructure from business applications, BlueField DPUs enhance data center security, streamline operations, and reduce total cost of ownership.

The BlueField DPU contains a programmable CPU based on Arm cores, a state-of-the-art NVIDIA® ConnectX®, and an enhanced set of security, storage, and networking accelerators that can be configured to perform multiple software-defined, hardware-accelerated functions. With a BlueField DPU, a software-defined network, and/or software-defined storage solution can be deployed and offloaded from the main host CPU in the server. Similarly, other dedicated services (e.g., distributed firewall, deep packet inspection, malware detection) can run on the BlueField DPU and can be accelerated with zero CPU overheads.

The BlueField DPU resembles a server embedded within the server itself, creating a secure environment where an infrastructure stack can operate independently from the primary (i.e., host) CPU, effectively isolating it from the untrusted tenant applications.

This is the recommended mode for utilizing the DPU in which software running on the host CPU has no direct access to the DPU. For instance, in a scenario where a cloud service provider is responsible for managing both networking and storage in a cloud infrastructure stack, it can establish an isolated environment within the DPU.

BlueField SuperNIC

The NVIDIA® BlueField® SuperNIC is the world's most advanced network accelerator, designed for supercharging hyperscale generative AI workloads. It delivers deterministic, isolated performance, with secure cloud multi-tenancy. Featured on the Spectrum-X networking platform, NVIDIA integrates BlueField-3 SuperNICs across its accelerated systems to enable peak AI workload efficiency. Powered by the NVIDIA DOCA software, the SuperNIC offers up to 400Gb/s connectivity between GPU servers, with features like RoCE adaptive routing, direct data placement (DDP), and programmable congestion control. With its unique HHL form factor and low-power platform, the BlueField-3 SuperNIC fits most enterprise-class servers.

Info

To read more about the BlueField-3 features and benefits, refer to [this page](#).

Info

To read more about the BlueField-2 features and benefits, refer to [this page](#).

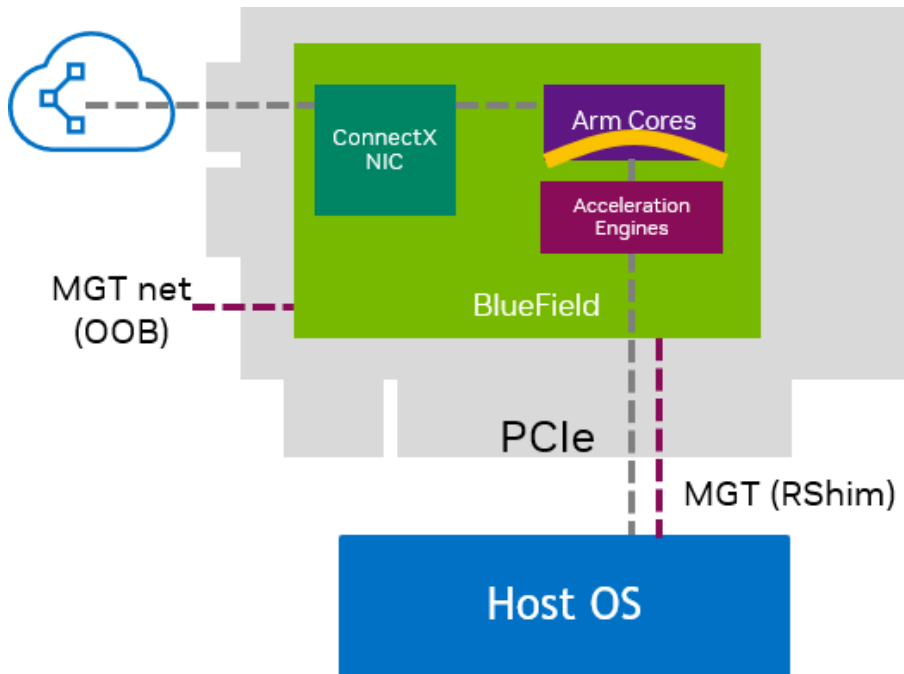
Operation Modes

The following subsections detail the available modes of operation.

DPU Mode

- Default mode for BlueField DPU
- Host-trusted mode
- Arm OS has embedded function (ECPF) ownership and controls the NIC's resources and data path

- BlueField controls and enforces network policies with the option of enforcing storage and security policies
- BlueField is the trusted function managed by the data center and host administrator



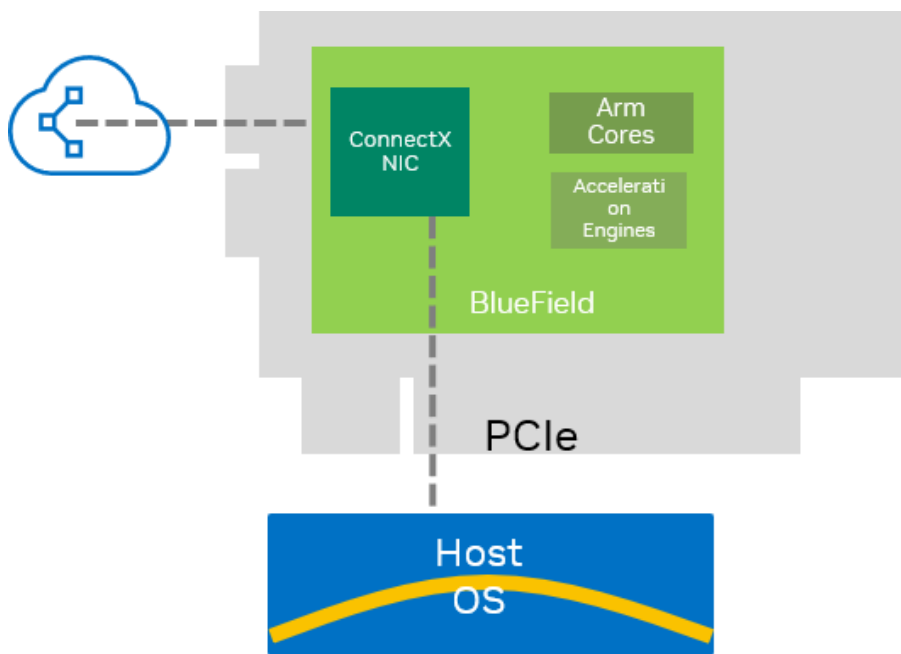
NIC Mode

- Default mode for BlueField SuperNIC
- BlueField behaves like a network adapter (NIC)
- Host is in full control of NIC functionality

(i) Info

All NIC offloads (as in NVIDIA® ConnectX® offloads) are enabled and available for the host

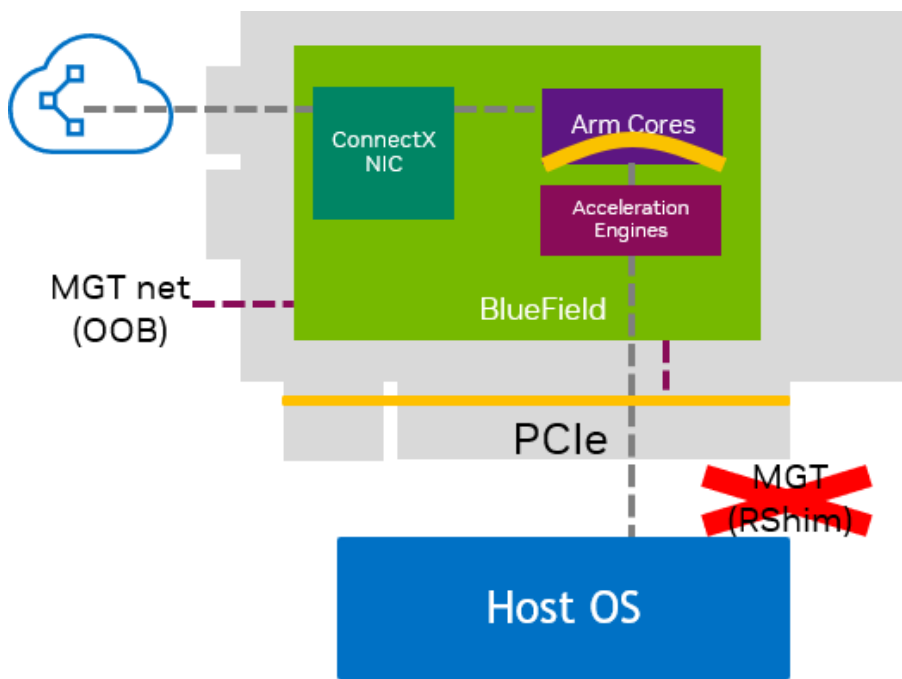
- BlueField Arm cores are halted and Arm OS stops running



Trust Model

Zero-trust Mode

- Arm OS has embedded function (ECPF) ownership and controls the NIC's resources and data path
- BlueField controls and enforces network policies with the option of enforcing storage and security policies
- Host is isolated; management from the host via PCIe edge connector is blocked
- Desired, safest state



(i) Note

By default, BlueField boots in trusted mode. Therefore, users must change the mode to zero-trust mode.

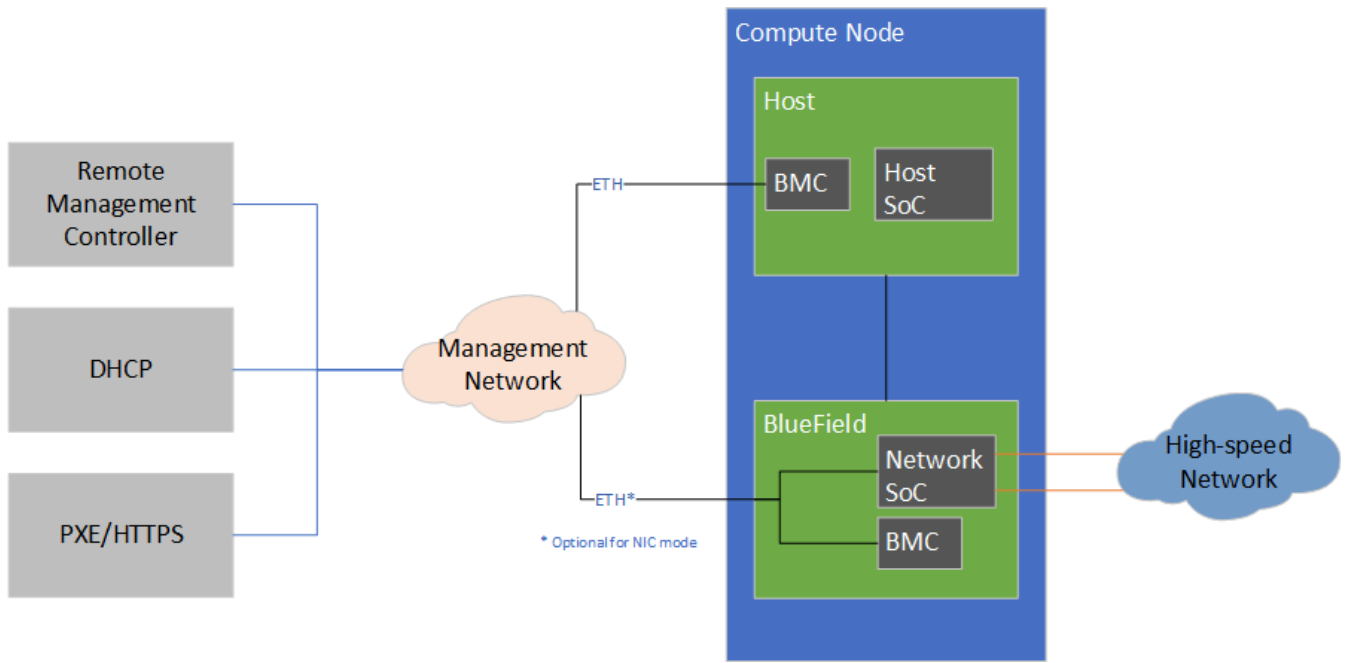
Prerequisites for Initial BlueField Deployment

The following is a list of system requirements for the deployment process:

- Connection to management network:
 - For BlueField DPU, BlueField BMC 1GbE interface connected to the management network via ToR
 - For BlueField SuperNIC, the host BMC connected to the management network via ToR or via NC-SI passthrough over data ports
 - BlueField BMC 1GbE interface connected to the management network via ToR is optional (MB312 use case)
- Remote management controller (RMC) connected to BlueField BMC or host BMC Ethernet interface via ToR
- DHCP server existing in the management network
- An NVQual certified server

Info

RMC is the platform for data center infrastructure managers to manage BlueFields from Top of Rack (ToR) over 1 GbE.



First-time Installation Procedure

The installation procedure depends on the active mode of operation on the NVIDIA® BlueField® networking platform:

- [DPU Mode installation](#) (default mode for BlueField DPUs)
- [NIC Mode installation](#) (default mode for BlueField SuperNICs)

DPU Mode Installation

Note

DPU mode is the default mode for BlueField DPUs, while BlueField SuperNICs are shipped with NIC mode as their default. To switch between the modes, see [NVIDIA BlueField Modes of Operation](#). To check which mode your BlueField is currently running:

- Using RedFish on the 1GbE interface, refer to [NVIDIA BlueField BMC Software Documentation](#)
- From the host server, refer to "[Enabling NIC Mode on BlueField-3 from Linux](#)"
- From the host server's UEFI BIOS menu, refer to "[Configuring NIC Mode on BlueField-3 from Host BIOS HII UEFI Menu](#)"

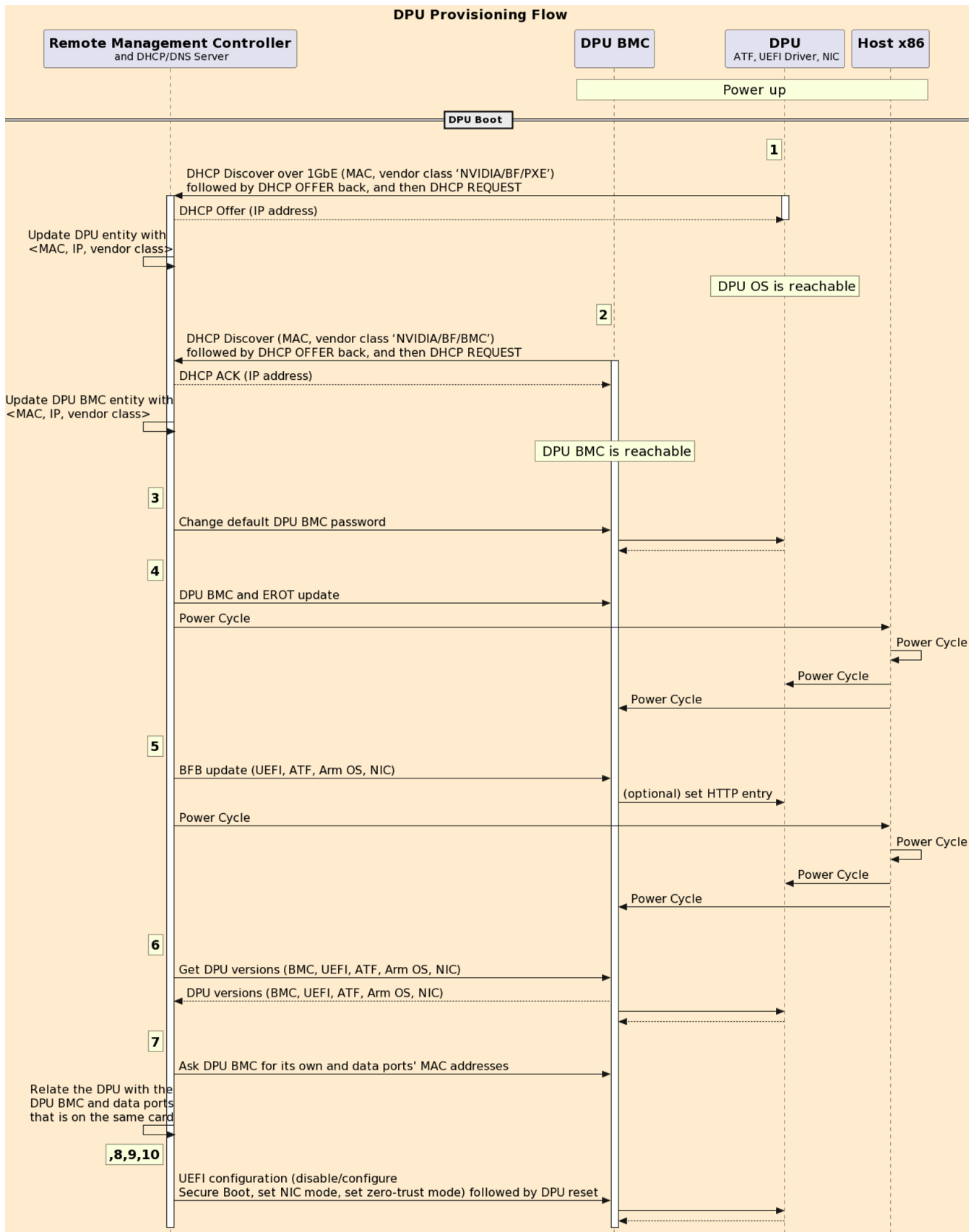
Note

In the out-of-box state of the BlueField the host is assumed to be trusted. Later in this procedure, after performing BFB Bundle update, a step is provided to disable the host RShim which the user must perform to protect the BlueField from potential security threats from the host.

The following diagram illustrates the sequence of events and actions from first time power-up of the NVIDIA® BlueField® networking platform (DPU or SuperNIC) in the data center environment through provisioning and maintenance.

Info

The numbers indicated in the sequence diagram correspond to the steps that follow it.



At the end of this procedure, the BlueField should be configured with an IP address, all required settings, has up-to-date software component versions, and is ready to use.

Step 1 – BlueField SoC Boots

The BlueField SoC boots to the UEFI BIOS and DHCP DISCOVER is sent

1. BlueField SoC runs UEFI/PXE which sends a DHCP DISCOVER over the 1GbE OOB interface, including vendor class ("NVIDIA/BF/PXE") for BlueField SoC (to allow customer's server to differentiate between BlueField SoC and BlueField BMC), and MAC for identification and discovery. See [Appendix B](#) for more information.
2. A customer's DHCP server inspects the MAC address and the vendor class, allocates IP, and continues the standard DHCP.
3. DHCP server updates RMC of the new BlueField discovered with detailed information (e.g., MAC, IP address, vendor class).

Step 2 – BlueField BMC Boots

BlueField BMC issues DHCP DISCOVER over the 1GbE OOB interface, including vendor class ("NVIDIA/BF/BMC") for BlueField-BMC, and MAC for identification and discovery.

Example of BlueField BMC DHCP DISCOVER packet structure (note "NVIDIA/BF/BMC" in line 13):

```
root@bf-bmc:~# 18:18:10.563269 IP (tos 0xc0, ttl 64, id 0, offset 0, flags [none], proto UDP (17), length 320)
0.0.0.0.bootpc > 255.255.255.255.bootps: [udp sum ok] BOOTP/DHCP, Request from b8:3f:d2:ca:4b:26
(oui Unknown), length 292, xid 0xfc2acdec, secs 1, Flags [none] (0x0000)
Client-Ethernet-Address b8:3f:d2:ab:cd:ef (oui Unknown)
Vendor-rfc1048 Extensions
Magic Cookie 0x63825363
DHCP-Message (53), length 1: Discover
Client-ID (61), length 7: ether b8:3f:d2:ab:cd:ef
Parameter-Request (55), length 9:
Subnet-Mask (1), Default-Gateway (3), Domain-Name-Server (6), Hostname (12)
Domain-Name (15), Static-Route (33), NTP (42), Unknown (120)
Classless-Static-Route (121)
MSZ (57), length 2: 576
Hostname (12), length 7: "bf-bmc" Vendor-Class (60), length 13: "NVIDIA/BF/BMC" END (255), length 0
18:18:10.565261 IP (tos 0x0, ttl 63, id 0, offset 0, flags [DF], proto UDP (17), length 353)
```

```

(example) dhcp01.XX.YY > Idev-platform-13-043-bmc.bootpc: [no cksum] BOOTP/DHCP, Reply, length
325, hops 1, xid 0xfc2acdec, secs 1, Flags [none] (0x0000)
(example) Your-IP Idev-platform-13-043-bmc.XX.YY
(example) Server-IP I-pxe02.XX.YY
Gateway-IP 10.237.0.255
Client-Ethernet-Address b8:3f:d2:ab:cd:ef (oui Unknown)
file "pxelinux.0" Vendor-rfc1048 Extensions
Magic Cookie 0x63825363
DHCP-Message (53), length 1: Offer
Server-ID (54), length 4: (example) dhcp01.XX.YY
Lease-Time (51), length 4: 43200
Subnet-Mask (1), length 4: 255.255.0.0
Default-Gateway (3), length 4
(example) GW.XX.YY
Hostname (12), length 24: "Idev-platform-13-043-bmc" Domain-Name (15), length 13: "<local domain
name>" NTP (42), length 4: (example) NTP.XX.YY
END (255), length 0
18:18:10.565261 IP (tos 0x0, ttl 62, id 0, offset 0, flags [DF], proto UDP (17), length 353)
dhcp01.XX.YY > Idev-platform-13-043-bmc.<local domain name>: [no cksum] BOOTP/DH

```

1. DHCP server inspects the MAC address and the vendor class, allocates IP and continues the standard DHCP flow.
2. DHCP server updates RMC of the new BlueField BMC discovered with detailed information: MAC, IP address, vendor classes, etc.

Step 3 – Change Default Password

To communicate with the BlueField BMC, change the default password (0penBmc) by sending the following Redfish schema to the BlueField BMC:

```

curl -k -u root:0penBmc -H "Content-Type: application/json" -X PATCH https://<BF-BMC-IP>/redfish/v1/AccountService/Accounts/root -d '{"Password" : "<user-password>"}'

```

Where <BF-BMC-IP> is the IP address for the BlueField BMC (e.g., 10.10.1.2), and <user-password> is the chosen password to log into the BlueField BMC with root privileges.

The BMC password must comply with the following policy parameters:

- Using ASCII and Unicode characters is permitted
- Minimum length: 12
- Maximum length: 20
- Maximum number of consecutive character pairs: 4

Info

Two characters are consecutive if $|\text{hex}(\text{char}_1) - \text{hex}(\text{char}_2)| = 1$.

Examples of passwords with 5 consecutive character pairs (invalid): DcBa123456AbCd!; ab1XbcYcdZdeGef!; Testing_123abcgh!.

The following is a valid example password:

- HelloNvidia3D!

Note

A user account is locked for 10 minutes after 10 consecutive failed attempts.

For example:

```
[redfish_scripts] $ curl -k -u root:OpenBmc -H "Content-Type: application/json" -X PATCH https://<BF-BMC-IP>/redfish/v1/AccountService/Accounts/root -d '{"Password": "HelloNvidia3D!"}'
Response:
{
  "@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
```

```
"Message": "The request completed successfully.",
"MessageArgs": [],
"MessageId": "Base.1.15.0.Success",
"MessageSeverity": "OK",
"Resolution": "None"
}
]
}
```

Step 4 – Upgrade BlueField BMC Firmware

Upgrade BlueField BMC firmware via the Redfish "update service schema" through the 1GbE OOB.

- If a BlueField-2 is in your possession and it is the first time you are upgrading BlueField BMC, follow [Appendix A](#).
- If a BlueField-3 is in your possession, follow the instructions in the following subsections

Info

Make sure to download the latest BlueField BMC image available from the [BlueField Runtime and Driver Downloader](#).

Update BMC Firmware

1. Run the following Redfish command over the 1GbE out-of-band interface on the BlueField BMC to trigger a secure BlueField BMC firmware update:

```
curl -k -u root:'<password>' -H "Content-Type: application/octet-stream" -X POST -T
<package_path> https://<BF-BMC-IP>/redfish/v1/UpdateService/update
```

Where:

- <password> – BlueField BMC password
- <package_path> – BMC firmware update package path pointing to BMC *.fwpkg binary (e.g., bf3-bmc-23.09-6_opn.fwpkg)
- <BF-BMC-IP> – BMC IP address

After pushing the image to the BlueField BMC, a new task is created. Example:

```
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/0",
  "@odata.type": "#Task.v1_4_3.Task",
  "Id": "0",
  "TaskState": "Running"
}
```

Info

BMC firmware update takes ~12 minutes.

2. To track the progress of the update, use the task id received in the response above (i.e., 0) in your query and monitor the value of the task's PercentComplete field:

```
curl -k -u root:'<password>' -X GET https://<BF-BMC-IP>/redfish/v1/TaskService/Tasks/<task_id> |
jq -r '.PercentComplete'
```

Where:

- <password> – BlueField BMC password
- <BF-BMC-IP> – BMC IP address
- <task_id> – task ID of the update process as received in the response under the Id value

Example output:

```
% Total  % Received % Xferd  Average Speed   Time    Time     Time  Current Dload
Upload  Total  Spent    Left  Speed
100 2123  100 2123   0    0 38600    0 --:--:-- --:--:-- --:--:-- 37910
20
```

See PercentComplete is at 20 percent.

3. Proceed to the next step when the process reaches 100%.

Update eROT Firmware

1. Trigger a secure firmware update:

```
curl -k -u root:'<password>' -H "Content-Type: application/octet-stream" -X POST -T
<package_path> https://<BF-BMC-IP>/redfish/v1/UpdateService/update
```

Where:

- <password> – BlueField BMC password
- <package_path> – eROT firmware update package path pointing to eROT *.fwpkg binary (e.g. cec1736-ecfw-00.02.0127.0000-n02-rel-prod.fwpkg)
- <BF-BMC-IP> – BMC IP address

After initiating the eROT secure update, a new task is created. Example:

```
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/0",
  "@odata.type": "#Task.v1_4_3.Task",
  "Id": "0",
  "TaskState": "Running"
```

```
}
```

Info

eROT firmware update takes ~20 seconds.

2. To track the progress of the update, use the task id received in the response above (i.e., 0) in your query and monitor the value of the task's PercentComplete field:

```
curl -k -u root:'<password>' -X GET https://<BF-BMC-IP>/redfish/v1/TaskService/Tasks/<task_id> |  
jq -r '.PercentComplete'
```

Note

Run this command several times until PercentComplete shows 100 before proceeding to other operations.

Where:

- o <password> – BlueField BMC password
- o <BF-BMC-IP> – BMC IP address
- o <task_id> – task ID of the update process as received in the response under the Id value

Note

For the firmware of the BMC and CEC to apply and to allow new Redfish APIs which are required for the following steps, a power cycle

of the BlueField is required. The BlueField-3 is installed in the host's PCIe slot. To initiate the power cycle sequence for the BlueField, the entire server on which it is installed must be power cycled.

Possible Error Codes During BMC/eROT Upgrade

Fault	Diagnosis and Possible Solution
<p>Connection to BMC breaks during firmware package transfer</p>	<ul style="list-style-type: none"> • Redfish task URI is not returned by the Redfish server • The Redfish server (if operational) is in idle state • After a reboot of BMC, or restart/recovery of the Redfish server, the Redfish server is in idle state <p>A new firmware update can be attempted by the Redfish client.</p>
<p>Connection to BMC breaks during firmware update</p>	<ul style="list-style-type: none"> • Redfish task URI previously returned by the Redfish server is no longer accessible • The Redfish server (if operational) is in one of the following states: <ul style="list-style-type: none"> ◦ In idle state, if the firmware update has completed ◦ In update state, if the firmware update is still ongoing • After a BMC reboot, or the restart/recovery of the Redfish server, the Redfish server is in idle state <p>A new firmware update can be attempted by the Redfish client.</p>
<p>Two firmware update requests are initiated</p>	<p>The Redfish server blocks the second firmware update request and returns the following:</p> <ul style="list-style-type: none"> • HTTP code 400 "Bad Request" • Redfish message based on standard registry entry UpdateInProgress <p>Check the status of the ongoing firmware update by looking at the TaskCollection resource.</p>
<p>Redfish task hangs</p>	<ul style="list-style-type: none"> • Redfish task URI that previously returned by the Redfish server is no longer accessible • PLDM-based firmware update progresses

Fault	Diagnosis and Possible Solution
	<ul style="list-style-type: none"> • After a reboot of BMC, or restart/recovery of the Redfish server, the Redfish server us in idle state <p>A new firmware update can be attempted by the Redfish client.</p>
BMC-EROT communication failure during image transfer	<p>The Redfish task monitoring the firmware update indicates a failure:</p> <ul style="list-style-type: none"> • TaskState is set to Exception • TaskStatus is set to Warning • Messages array in the task includes an entry based on the standard registry Update.1.0.0.TransferFailed indicating the components that failed during image transfer <p>The Redfish client may retry the firmware update.</p>
Firmware update fails	<p>The Redfish task monitoring the firmware update indicates a failure:</p> <ul style="list-style-type: none"> • TaskState is set to Exception • TaskStatus is set to Warning • Messages array in the task includes an entry describing the error <p>The Redfish client may retry the firmware update.</p>
ERoT failure (not responding)	<p>The Redfish task monitoring the firmware update indicates a failure:</p> <ul style="list-style-type: none"> • TaskState is set to Canceled • TaskStatus is set to Warning • Messages array in the task includes an entry describing the error • The Redfish client reports the error <p>The Redfish client may retry the firmware update.</p>
Firmware image validation failure	<p>The Redfish task monitoring the firmware update indicates a failure:</p> <ul style="list-style-type: none"> • TaskState is set to Exception • TaskStatus is set to Warning

Fault	Diagnosis and Possible Solution
	<ul style="list-style-type: none"> Messages array in the task includes an entry based on the standard registry Update.1.0.0.VerificationFailed to indicate the component for which verification failed The Redfish client reports the error <p>The Redfish client might retry the firmware update.</p>
Power loss before activation command is sent	<ul style="list-style-type: none"> The Redfish server is in idle state <p>A new firmware update can be attempted by the Redfish client.</p>
Firmware activation failure	<p>The Redfish task monitoring the firmware update indicates a failure:</p> <ul style="list-style-type: none"> TaskState is set to Exception TaskStatus is set to Warning Messages array in the task includes an entry based on the standard registry Update.1.0.ActivateFailed <p>The Redfish client may retry the firmware update.</p>
Push to BMC firmware package greater than 200 MB	<ul style="list-style-type: none"> No Redfish task is created Messages array in the task includes an entry based on the standard registry Base.1.8.1.ResourceExhaustion and a request to retry the operation is given

Step 5 – Upgrade BlueField Firmware Components and BSP

Upgrade the BlueField firmware components (i.e., ATF, UEFI, NIC-firmware) and the BSP using the BFB image.

Info

Make sure to download the latest DOCA image (BFB file) available from the [BlueField Runtime and Driver Downloader](#).

The BFB installation procedure consists of the following main stages:

1. Disabling RShim on the server.
2. Initiating the BFB update procedure by transferring the BFB image using one of the following options:
 - [Redfish interface](#) – SimpleUpdate with [SCP](#), [HTTP](#), or [HTTPS](#)
 1. Confirming the identity of the host and BMC—required only for SCP, during first-time setup or after BMC factory reset.
 2. Sending a SimpleUpdate request.
 - [Direct SCP](#)
3. [Tracking the installation's progress and status](#).

Note

While the BlueField Bundle (BFB) contains NIC firmware images, it does not automatically install them. To automatically install the NIC firmware during BFB upgrade, generate the configuration file `bf.cfg` and combine it with the BFB file:

```
# echo WITH_NIC_FW_UPDATE=yes > bf.cfg
# cat <path_to_bfb> bf.cfg > new.bfb
```

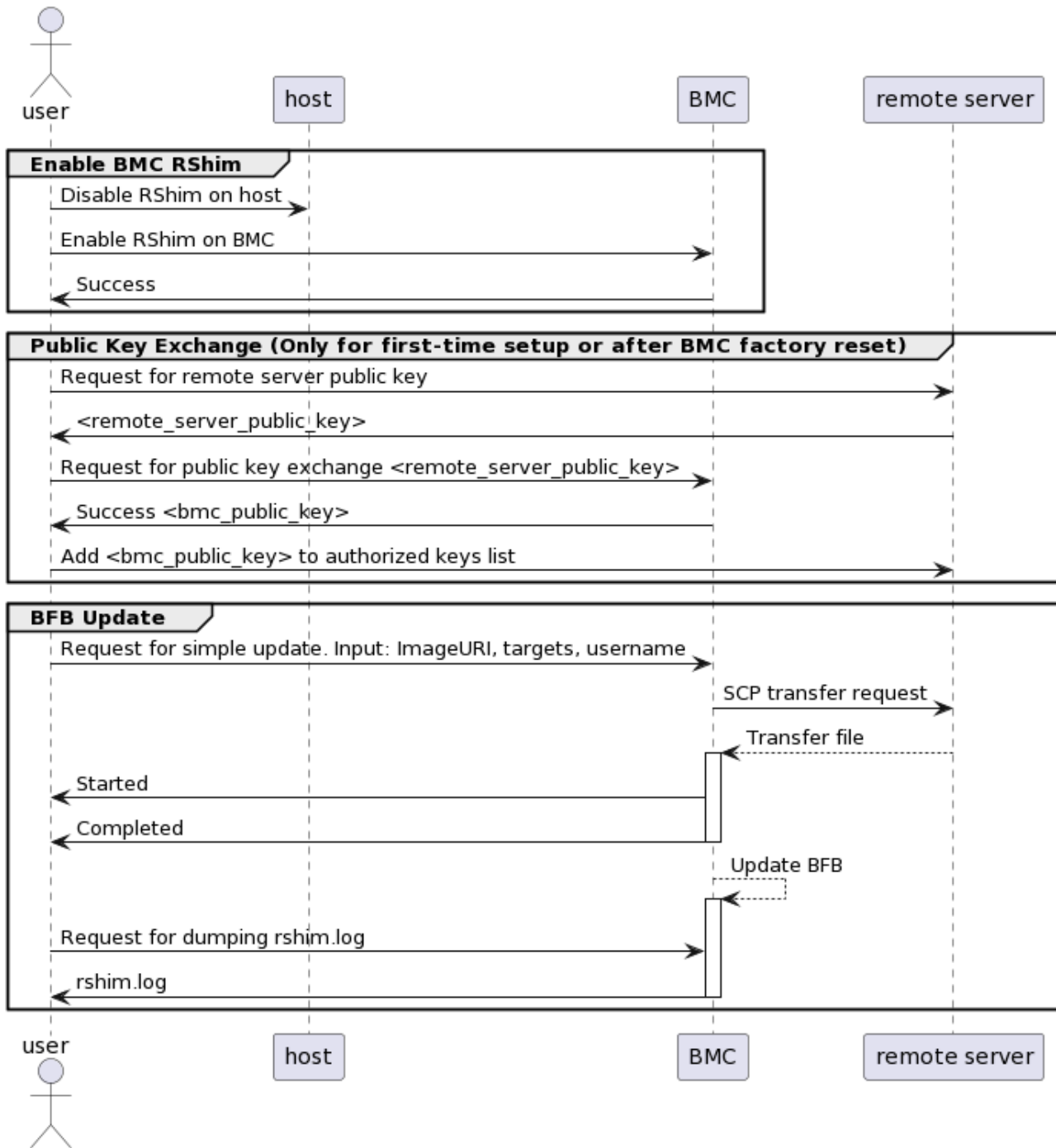
Transferring BFB File

Since the BFB is too large to store on the BMC flash or tmpfs, the image must be written to the RShim device. This can be done by either running SCP directly or using the Redfish interface.

Redfish Interface

Installing BFB File Using SCP Protocol

BMC Image Update Flow Using UpdateService POST Command



The following are the detailed instructions outlining each step in the diagram above:

1. Prepare secure file transfer of BFB:

Note

The following is an example for how to generate the server public key on Ubuntu 22.04 and it may be different on other OS distributions/versions.

1. Gather the public SSH host keys of the server holding the `new.bfb` file. Run the following against the server holding the `new.bfb` file ("Remote Server"):

i Info

OpenSSH is required for this step.

```
ssh-keyscan -t <key_type> <remote_server_ip>
```

Where:

- `key_type` – the type of key associated with the server storing the BFB file (e.g., `ed25519`)
 - `remote_server_ip` – the IP address of the server hosting the BFB file
2. Retrieve the remote server's public key from the response, and send the following Redfish command to the BlueField BMC:

```
curl -k -u root:<password> -H "Content-Type: application/json" -X POST -d  
'{"RemoteServerIP": "<remote_server_ip>", "RemoteServerKeyString": "  
<remote_server_public_key>"}'  
https://<bmc_ip>/redfish/v1/UpdateService/Actions/Oem/NvidiaUpdateService.PublicKeyExc
```

Where:

- `password` – BlueField BMC password

- remote_server_ip – the IP address of the server hosting the BFB file
- remote_server_public_key – remote server's public key from the ssh-keyscan response, which contains both the type and the public key with **one space** between the two fields (i.e., "<type> <public_key>")
- bmc_ip – BMC IP address

3. Extract the BMC public key information (i.e., "<type> <bmc_public_key> <username>@<hostname>") from the PublicKeyExchange response and append it to the authorized_keys file on the remote server holding the BFB file. This enables password-less key-based authentication for users.

```
{
  "@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "Please add the following public
key info to ~/.ssh/authorized_keys on the
remote server",
      "MessageArgs": [
        "<type> <bmc_public_key> root@dpu-bmc"
      ]
    },
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The request completed
successfully.",
      "MessageArgs": [],
      "MessageId": "Base.1.15.0.Success",
      "MessageSeverity": "OK",
      "Resolution": "None"
    }
  ]
}
```

2. Initiate image transfer. Run the following Redfish command:

```
curl -k -u root:<password> -H "Content-Type: application/json" -X POST -d
{"TransferProtocol":"SCP", "ImageURI":"<image_uri>","Targets":
```

```
["redfish/v1/UpdateService/FirmwareInventory/DPU_OS"], "Username":"<username>"}'  
https://<bmc_ip>/redfish/v1/UpdateService/Actions/UpdateService.SimpleUpdate
```

Note

This command uses SCP for the image transfer, initiates a soft reset on the BlueField, and then pushes the boot stream. For NVIDIA-supplied BFBs, the eMMC is flashed automatically once the boot stream is pushed. Upon success, a running message is received.

Info

After the BMC boots, it may take a few seconds (6-8 seconds for NVIDIA® BlueField®-2, and 2 seconds for BlueField-3) until the BlueField BSP (DPU_OS) is up.

Where:

- `image_uri` – contains both the remote server IP address and the full path to the `.bfb` file on the remote server, with **one slash** between the two fields (i.e., `<remote_server_ip>/<full_path_of_bfb>`).

Info

For example, if `<remote_server_ip>` is `10.10.10.10` and `<full_path_of_bfb>` is `/tmp/file.bfb` then `"ImageURI":"10.10.10.10//tmp/file.bfb"`.

- username – username on the remote server
- bmc_ip – BMC IP address

Response/error messages:

- If RShim is disabled:

```
{
  "error": {
    "@Message.ExtendedInfo": [
      {
        "@odata.type": "#Message.v1_1_1.Message",
        "Message": "The requested resource of type Target named '/dev/rshim0/boot'
was not found.",
        "MessageArgs": [
          "Target",
          "/dev/rshim0/boot"
        ],
        "MessageId": "Base.1.15.0.ResourceNotFound",
        "MessageSeverity": "Critical",
        "Resolution": "Provide a valid resource identifier and resubmit the request."
      }
    ],
    "code": "Base.1.15.0.ResourceNotFound",
    "message": "The requested resource of type Target named '/dev/rshim0/boot'
was not found."
  }
}
```

- If a username or any other required field is missing:

```
{
  "Username@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The create operation failed because the required property
Username was missing from the request.",
      "MessageArgs": [
        "Username"
      ],
    }
  ]
}
```

```
"MessageId": "Base.1.15.0.CreateFailedMissingReqProperties",
"MessageSeverity": "Critical",
"Resolution": "Correct the body to include the required property with a valid
value and resubmit the request if the operation failed."
}
]
}
```

- Success message if the request is valid and a task is created:

```
{
"@odata.id":
"/redfish/v1/TaskService/Tasks/<task_id>",
"@odata.type": "#Task.v1_4_3.Task",
"Id": "<task_id>",
"TaskState": "Running",
"TaskStatus": "OK"
}
```

3. Run the following Redfish command to track the SCP image's transfer status (percentage is not updated until it reaches 100%):

```
curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/TaskService/Tasks/<task_id>
```

Note

During the transfer, the PercentComplete value remains at 0. If no errors occur, the TaskState is set to Running, and a keep-alive message is generated every 5 minutes with the content "Transfer is still in progress (X minutes elapsed). Please wait". Once the transfer is completed, the PercentComplete is set to 100, and the TaskState is updated to Completed.

Upon failure, a message is generated with the relevant resolution.

Where:

- - bmc_ip – BMC IP address
 - task_id – task ID received by the UpdateService command response

Examples:

- - - Response/error messages:
 - If host identity is not confirmed or the provided host key is wrong:

```
{
  "@odata.type": "#MessageRegistry.v1_4_1.MessageRegistry",
  "Message": "Transfer of image '<file_name>' to '/dev/rshim0/boot' failed.",
  "MessageArgs": [
    "<file_name>",
    "/dev/rshim0/boot"
  ],
  "MessageId": "Update.1.0.TransferFailed",
  "Resolution": " Unknown Host: Please provide server's public key using
PublicKeyExchange ",
  "Severity": "Critical"
}
...
"PercentComplete": 0,
"StartTime": "<start_time>",
"TaskMonitor": "/redfish/v1/TaskService/Tasks/<task_id>/Monitor",
"TaskState": "Exception",
"TaskStatus": "Critical"
```

Info

In this case, revoke the remote server key using the following Redfish command:

```
curl -k -u root:<password> -H "Content-Type: application/json" -X POST -d '{"RemoteServerIP": "<remote_server_ip>"}' https://<bmc_ip>/redfish/v1/UpdateService/Actions/Oem/NvidiaUpdate
```

Where:

- remote_server_ip – remote server's IP address
- bmc_ip – BMC IP address

Then repeat steps 1 and 2.

- - - If the BMC identity is not confirmed:

```
{
  "@odata.type": "#MessageRegistry.v1_4_1.MessageRegistry",
  "Message": "Transfer of image '<file_name>' to '/dev/rshim0/boot' failed.",
  "MessageArgs": [
    "<file_name>",
    "/dev/rshim0/boot"
  ],
  "MessageId": "Update.1.0.TransferFailed",
  "Resolution": "Unauthorized Client: Please use the PublicKeyExchange action to receive the system's public key and add it as an authorized key on the remote server",
}
```

```

    "Severity": "Critical"
  }
  ...
  "PercentComplete": 0,
  "StartTime": "<start_time>",
  "TaskMonitor": "/redfish/v1/TaskService/Tasks/<task_id>/Monitor",
  "TaskState": "Exception",
  "TaskStatus": "Critical"

```

Info

In this case, verify that the BMC key has been added correctly to the `authorized_key` file on the remote server.

-
-
-

■ If SCP fails:

```

{
  "@odata.type": "#MessageRegistry.v1_4_1.MessageRegistry",
  "Message": "Transfer of image '<file_name>' to '/dev/rshim0/boot' failed.",
  "MessageArgs": [
    "<file_name>",
    "/dev/rshim0/boot"
  ],
  "MessageId": "Update.1.0.TransferFailed",
  "Resolution": "Failed to launch SCP",
  "Severity": "Critical"
}
...
"PercentComplete": 0,
"StartTime": "<start_time>",
"TaskMonitor": "/redfish/v1/TaskService/Tasks/<task_id>/Monitor",
"TaskState": "Exception",

```

```
"TaskStatus": "Critical"
```

- - -
- Success/status messages:
 - The keep-alive message:

```
{
  "@odata.type": "#MessageRegistry.v1_4_1.MessageRegistry",
  "Message": " <file_name>' is being transferred to
'/dev/rshim0/boot'.",
  "MessageArgs": [
    " <file_name>",
    "/dev/rshim0/boot"
  ],
  "MessageId": "Update.1.0.TransferringToComponent",
  "Resolution": "Transfer is still in progress (5 minutes elapsed):
Please wait",
  "Severity": "OK"
}
...
"PercentComplete": 0,
"StartTime": "<start_time>",
"TaskMonitor": "/redfish/v1/TaskService/Tasks/<task_id>/Monitor",
"TaskState": "Running",
"TaskStatus": "OK"
```

- Upon successful completion of SCP BFB transfer:

```
{
  "@odata.type": "#MessageRegistry.v1_4_1.MessageRegistry",
  "Message": "Device 'DPU' successfully updated with image
'<file_name>'.",
  "MessageArgs": [
    "DPU",
    "<file_name>"
  ]
}
```

```
    ],  
    "MessageId": "Update.1.0.UpdateSuccessful",  
    "Resolution": "None",  
    "Severity": "OK"  
  },  
  ...  
  "PercentComplete": 100,  
  "StartTime": "<start_time>",  
  "TaskMonitor": "/redfish/v1/TaskService/Tasks/<task_id>/Monitor",  
  "TaskState": "Completed",  
  "TaskStatus": "OK"
```

Installing BFB File with HTTP Protocol

1. Make sure the BFB file, `new.bfb`, is available on HTTP server
2. Initiate image transfer. Run the following Redfish command:

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST -d  
'{"TransferProtocol":"HTTP", "ImageURI":"<image_uri>","Targets":  
["redfish/v1/UpdateService/FirmwareInventory/DPU_OS"]}'  
https://<bmc_ip>/redfish/v1/UpdateService/Actions/UpdateService.SimpleUpdate
```

Note

This command uses HTTP to download the image, initiates a soft reset on the BlueField, and pushes the boot stream. For NVIDIA-supplied BFBs, the eMMC is flashed automatically once the boot stream is pushed. Upon success, a running message is received.

Info

After the BMC boots, it may take a few seconds (6-8 seconds in BlueField-2 and 2 seconds in BlueField-3) until the BlueField BSP (DPU_OS) is up.

Where:

-

- image_uri – contains both the HTTP server address and the exported path to the .bfb file on the server, with **one slash** between the two fields (i.e., <http_server>/<exported_path_of_bfb>).

Info

For example, if <http_server> is 10.10.10.10 and <exported_path_of_bfb> is /tmp/new.bfb then "ImageURI":"10.10.10.10//tmp/new.bfb".

- bmc_ip – BMC IP address

Response/error messages:

- If RShim is disabled:

```
{
  "error": {
    "@Message.ExtendedInfo": [
      {
        "@odata.type": "#Message.v1_1_1.Message",
        "Message": "The requested resource of type Target named '/dev/rshim0/boot'
was not found.",
        "MessageArgs": [
          "Target",
          "/dev/rshim0/boot"
        ],
      }
    ]
  }
}
```



```

    "MessageId": "Base.1.15.0.ResourceNotFound",
    "MessageSeverity": "Critical",
    "Resolution": "Provide a valid resource identifier and resubmit the request."
  }
],
"code": "Base.1.15.0.ResourceNotFound",
"message": "The requested resource of type Target named '/dev/rshim0/boot'
was not found."
}

```

- If the HTTPS server address is wrong or the HTTPS service is not stated, an "Unknown Host" error is expected:

```

{
  "@odata.type": "#MessageRegistry.v1_4_1.MessageRegistry",
  "Message": "Transfer of image 'new.bfb' to '/dev/rshim0/boot' failed.",
  "MessageArgs": [
    "new.bfb",
    "/dev/rshim0/boot"
  ],
  "MessageId": "Update.1.0.TransferFailed",
  "Resolution": "Unknown Host: Please provide server's public key using
PublicKeyExchange (for SCP download) or Check and restart server's web service
(for HTTP/HTTPS download)",
  "Severity": "Critical"
},

```

- If TransferProtocol or any other required field are wrong:

```

{
  "@Message.ExtendedInfo": [ {
    "@odata.type": "#Message.v1_1_1.Message",
    "Message": "The parameter TransferProtocol for the action
UpdateService.SimpleUpdate is not supported on the target resource.",
    "MessageArgs": [
      "TransferProtocol",
      "UpdateService.SimpleUpdate"
    ],
    "MessageId": "Base.1.16.0.ActionParameterNotSupported",

```

```

    "MessageSeverity": "Warning",
    "Resolution": "Remove the parameter supplied and resubmit the request if the
operation failed."
  }
]
}

```

- If Targets or any other required field are missing:

```

{
  "Targets@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The create operation failed because the required property Targets
was missing from the request.",
      "MessageArgs": [
        "Targets"
      ],
      "MessageId": "Base.1.16.0.CreateFailedMissingReqProperties",
      "MessageSeverity": "Critical",
      "Resolution": "Correct the body to include the required property with a valid
value and resubmit the request if the operation failed."
    }
  ]
}

```

- Success message if the request is valid and a task is created:

```

{
  "@odata.id":
"/redfish/v1/TaskService/Tasks/<task_id>",
  "@odata.type": "#Task.v1_4_3.Task",
  "Id": "<task_id>",
  "TaskState": "Running",
  "TaskStatus": "OK"
}

```

Installing BFB File with HTTPS Protocol

1. Make sure the BFB file, `new.bfb`, is available on HTTPS server
2. Make sure the BMC has certificate to authenticate the HTTPS server. Or install a valid certificate to authenticate:

```
curl -c cjar -b cjar -k -u root:'<password>' -X POST  
https://$bmc/redfish/v1/Managers/Bluefield_BMC/Truststore/Certificates -d @CAcert.json
```

3. Initiate image transfer. Run the following Redfish command:

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST -d  
'{"TransferProtocol":"HTTPS", "ImageURI":"<image_uri>","Targets":  
["redfish/v1/UpdateService/FirmwareInventory/DPU_OS"]}'  
https://<bmc_ip>/redfish/v1/UpdateService/Actions/UpdateService.SimpleUpdate
```

Note

This command uses HTTPS for the image download, initiates a soft reset on the BlueField, and then pushes the boot stream. For NVIDIA-supplied BFBs, the eMMC is flashed automatically once the boot stream is pushed. Upon success, a running message is received.

Info

After the BMC boots, it may take a few seconds (6-8 seconds in BlueField-2 and 2 seconds in BlueField-3) until the BlueField BSP (DPU_OS) is up.

Where:

- - image_uri – contains both the HTTPS server address and the exported path to the .bfb file on the server, with **one slash** between the two fields (i.e., <https_server>/<exported_path_of_bfb>).

Info

For example, if <https_server> is urm.nvidia.com and <exported_path_of_bfb> is `artifactory/sw-mlnx-bluefield-generic/Ubuntu22.04/new.bfb` then
"ImageURI":"10.126.206.42/artifactory/sw-mlnx-bluefield-generic/Ubuntu22.04/new.bfb".

- bmc_ip – BMC IP address

Response / error messages:

- - - If RShim is disabled:

```
{
  "error": {
    "@Message.ExtendedInfo": [
      {
        "@odata.type": "#Message.v1_1_1.Message",
        "Message": "The requested resource of type Target named '/dev/rshim0/boot'
was not found.",
        "MessageArgs": [
          "Target",
          "/dev/rshim0/boot"
        ],
        "MessageId": "Base.1.15.0.ResourceNotFound",
        "MessageSeverity": "Critical",
```

```

    "Resolution": "Provide a valid resource identifier and resubmit the request."
  }
],
"code": "Base.1.15.0.ResourceNotFound",
"message": "The requested resource of type Target named '/dev/rshim0/boot'
was not found."
}

```

- If the HTTPS server address is wrong or the HTTPS service is not stated, an "Unknown Host" error is expected:

```

{
  "@odata.type": "#MessageRegistry.v1_4_1.MessageRegistry",
  "Message": "Transfer of image 'new.bfb' to '/dev/rshim0/boot' failed.",
  "MessageArgs": [
    "new.bfb",
    "/dev/rshim0/boot"
  ],
  "MessageId": "Update.1.0.TransferFailed",
  "Resolution": "Unknown Host: Please provide server's public key using
PublicKeyExchange (for SCP download) or Check and restart server's web service
(for HTTP/HTTPS download)",
  "Severity": "Critical"
},

```

- If TransferProtocol or any other required field are wrong:

```

{
  "@Message.ExtendedInfo": [ {
    "@odata.type": "#Message.v1_1_1.Message",
    "Message": "The parameter TransferProtocol for the action
UpdateService.SimpleUpdate is not supported on the target resource.",
    "MessageArgs": [
      "TransferProtocol",
      "UpdateService.SimpleUpdate"
    ],
    "MessageId": "Base.1.16.0.ActionParameterNotSupported",
    "MessageSeverity": "Warning",
  }
],

```

```
    "Resolution": "Remove the parameter supplied and resubmit the request if the
operation failed."
  }
]
}
```

- -

- If Targets or any other required field are missing:

```
{
  "Targets@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The create operation failed because the required property Targets
was missing from the request.",
      "MessageArgs": [
        "Targets"
      ],
      "MessageId": "Base.1.16.0.CreateFailedMissingReqProperties",
      "MessageSeverity": "Critical",
      "Resolution": "Correct the body to include the required property with a valid
value and resubmit the request if the operation failed."
    }
  ]
}
```

- -

- If the HTTPS server fails to authenticate the current installed certificate:

```
{
  "@odata.type": "#MessageRegistry.v1_4_1.MessageRegistry",
  "Message": "Transfer of image 'new.bfb' to '/dev/rshim0/boot' failed.",
  "MessageArgs": [
    "new.bfb",
    "/dev/rshim0/boot"
  ]
}
```

```
    ],
    "MessageId": "Update.1.0.TransferFailed",
    "Resolution": "Bad Certificate: Please check the remote server certification,
correct and replace the current installed one",
    "Severity": "Critical"
  },
```

-

-

- Success message if the request is valid and a task is created:

```
{
  "@odata.id":
  "/redfish/v1/TaskService/Tasks/<task_id>",
  "@odata.type": "#Task.v1_4_3.Task",
  "Id": "<task_id>",
  "TaskState": "Running",
  "TaskStatus": "OK"
}
```

Tracking Image Transfer Status and Progress for HTTP/HTTPS Protocols

The following section is relevant for HTTP/HTTPS protocols which received a success message of a valid SimpleUpdate request and a running task state.

Run the following Redfish command to track image transfer status and progress:

```
curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/TaskService/Tasks/<task_id>
```

Example:

```
{
  "@odata.type": "#MessageRegistry.v1_4_1.MessageRegistry",
  "Message": "Image 'new.bfb' is being transferred to '/dev/rshim0/boot'.",
```

```
"MessageArgs": [  
  "new.bfb",  
  "/dev/rshim0/boot"  
],  
"MessageId": "Update.1.0.TransferringToComponent",  
"Resolution": "Transfer started",  
"Severity": "OK"  
},  
...  
  
"PercentComplete": 60,  
"StartTime": "2024-06-10T19:39:03+00:00",  
"TaskMonitor": "/redfish/v1/TaskService/Tasks/1/Monitor",  
"TaskState": "Running",  
"TaskStatus": "OK"
```

Direct SCP

```
scp <path_to_bfb> root@<bmc_ip>:/dev/rshim0/boot
```

If bf.cfg is required as part of the boot process, run:

```
cat <path_to_bfb> bf.cfg > new.bfb  
scp <path to new.bfb> root@<bmc_ip>:/dev/rshim0/boot
```

Tracking Installation Progress and Status

After image transfer is complete, users may follow the installation progress and status with the help of a dump of current the RShim miscellaneous messages log.

1. Initiate request for dump download:


```
sudo curl -k -u root:'<password>' -d '{"DiagnosticDataType": "Manager"}' -X POST
https://<ip_address>/redfish/v1/Managers/Bluefield_BMC/LogServices/Dump/Actions/LogService.Co
```

Where:

- <ip-address> – BMC IP address
- <password> – BMC password

2. Use the received task ID to poll for dump completion:

```
sudo curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET
https://<ip_address>/redfish/v1/TaskService/Tasks/<task_id>
```

Where:

- <ip-address> – BMC IP address
- <password> – BMC password
- <task_id> – Task ID received from the first command

3. Once dump is complete, download and review the dump:

```
sudo curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET
https://<ip_address>/redfish/v1/Managers/Bluefield_BMC/LogServices/Dump/Entries/<entry_id>/at
--output </path/to/tar/log_dump.tar.xz>
```

Where:

- <ip-address> – BMC IP address
- <password> – BMC password
- <entry_id> – The entry ID of the dump in
redfish/v1/Managers/Bluefield_BMC/LogServices/Dump/Entries

- o `</path/to/tar/log_dump.tar.xz>` – path to download the log dump `log_dump.tar.xz`

4. Untar the file to review the logs. For example:

```
tar xvfj log_dump.tar.xz
```

5. The log is contained in the `rshim.log` file. The log displays Reboot, finished, DPU is ready, or In Enhanced NIC mode when BFB installation completes.

Note

If the downloaded log file does not contain any of these strings, keep downloading the log file until they appear.

6. When installation is complete, you may crosscheck the new BFB version against the version provided to verify a successful upgrade:

```
curl -k -u root:"<PASSWORD>" -H "Content-Type: application/json" -X GET  
https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory/DPU_OS
```

Example response:

```
"@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_OS",  
"@odata.type": "#SoftwareInventory.v1_4_0.SoftwareInventory",  
"Description": "Host image",  
"Id": "DPU_OS",  
"Members@odata.count": 1,  
"Name": "Software Inventory",  
"RelatedItem": [  
  {  
    "@odata.id": "/redfish/v1/Systems/Bluefield/Bios"  
  }  
],
```

```

"SoftwareId": "",
"Status": {
  "Conditions": [],
  "Health": "OK",
  "HealthRollup": "OK",
  "State": "Enabled"
},
"Updateable": true,
"Version": "DOCA_2.2.0_BSP_4.2.1_Ubuntu_22.04-8.23-07"

```

Step 6 – Verify Software Component Versions

Verify BlueField BSP, BlueField BMC and BlueField NIC firmware versions are up to date according to the [NVIDIA BlueField BMC Software User Manual](#) and [NVIDIA BlueField BSP Release Notes](#).

1. Use the Redfish FirmwareInventory schema over the 1GbE OOB interface to the BlueField's BMC:

```

[redfish_scripts] $ curl -k -u root:<password> -H "Content-Type: application/octet-stream" -X GET
https://<BF-BMC-IP>/redfish/v1/UpdateService/FirmwareInventory
{
  "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory",
  "@odata.type": "#SoftwareInventoryCollection.SoftwareInventoryCollection",
  "Members": [
    {
      "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/9f7ec75a_BMC_Firmware"
    },
    {
      "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/Bluefield_FW_ERoT"
    },
    {
      "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_ATF"
    },
    {
      "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_BOARD"
    },
    {
      "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_BSP"
    }
  ]
}

```

```

    },
    {
      "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_NIC"
    },
    {
      "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_NODE"
    },
    {
      "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_OFED"
    },
    {
      "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_OS"
    },
    {
      "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_SYS_IMAGE"
    },
    {
      "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_UEFI"
    }
  ],
  "Members@odata.count": 11,
  "Name": "Software Inventory Collection"
}

```

Response example for DPU_ATF:

```

> curl -k -u root:<password> -H "Content-Type: application/octet-stream" -X GET https://<BF-
BMC-IP>/redfish/v1/UpdateService/FirmwareInventory/DPU_ATF
{
  "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_ATF",
  "@odata.type": "#SoftwareInventory.v1_4_0.SoftwareInventory",
  "Description": "Host image",
  "Id": "DPU_ATF",
  "Members@odata.count": 1,
  "Name": "Software Inventory",
  "RelatedItem": [
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/Bios"
    }
  ],
  "SoftwareId": "",
  "Status": {

```

```
"Health": "OK",
"HealthRollup": "OK",
"State": "OK",
},
"Updateable": true,
"Version": "v2.2(release):4.0.2-33-gd9f4ad5"
```

Info

This request may also be used to query some of the other previously mentioned components (e.g., `9f7ec75a_BMC_Firmware`, `Bluefield_FW_ERoT`).

2. If the versions are not as expected, upgrade as needed:

1. Download the latest DOCA (BFB file) versions from the downloader at the bottom of the [DOCA product page](#).
2. DOCA (BFB) upgrade options (upgrading UEFI, ATF, Arm OS, NIC firmware components):
 - Recommended—BFB upgrade from remote management controller using Redfish UpdateService schema over 1GbE to BlueField BMC:

```
export token=`curl -k -H "Content-Type: application/json" -X POST
https://<bmc_ip>/login -d '{"username":"root","password":"<password>"}' | grep
token | awk '{print $2;}' | tr -d "'`
```

For more information on deploying BlueField software from the BMC, refer to the "Deploying BlueField Software Using BFB from BMC" page of the [NVIDIA BlueField BSP](#) document.

Step 7 – Relate BlueField to BlueField BMC and NIC Data Ports on Same Machine

1. Get the BlueField's BMC MAC address using the following Redfish command over the 1GbE OOB port to the BlueField BMC:

```
curl -k -u root:<password> -H 'Content-Type: application/json' -X GET https://<BF-BMC-IP>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0
{
  "@odata.id": "/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0",
  "@odata.type": "#EthernetInterface.v1_6_0.EthernetInterface",
  "DHCPv4": {
    "DHCPEnabled": true,
    "UseDNSServers": true,
    "UseDomainName": true,
    "UseNTPServers": true
  },
  "DHCPv6": {
    "OperatingMode": "Stateful",
    "UseDNSServers": true,
    "UseDomainName": true,
    "UseNTPServers": true
  },
  "Description": "Management Network Interface",
  "FQDN": "dpu-bmc",
  "HostName": "BlueField-bmc",
  "IPv4Addresses": [
    {
      "Address": "10.237.40.179",
      "AddressOrigin": "DHCP",
      "Gateway": "0.0.0.0",
      "SubnetMask": "255.255.0.0"
    }
  ],
  "IPv4StaticAddresses": [],
  "IPv6AddressPolicyTable": [],
  "IPv6Addresses": [
    {
      "Address": "fdfd:fdfd:10:237:966d:aeff:fe17:9f5f",
      "AddressOrigin": "DHCPv6",
      "AddressState": null,
      "PrefixLength": 64
    },
    {
      "Address": "fe80::966d:aeff:fe17:9f5f",
      "AddressOrigin": "LinkLocal",
```

```

    "AddressState": null,
    "PrefixLength": 64
  }
],
"IPv6DefaultGateway": "fe80::445b:ed80:5f97:8900",
"IPv6StaticAddresses": [],
"Id": "eth0",
"InterfaceEnabled": true,
"LinkStatus": "LinkUp",
"MACAddress": "94:6d:ae:17:9f:5f",
"MTUSize": 1500,
"Name": "Manager Ethernet Interface",
"NameServers": [
  "fdfd:fdfd:7:77:250:56ff:fe8b:e4f9"
],
"SpeedMbps": 0,
"StaticNameServers": [],
"Status": {
  "Health": "OK",
  "HealthRollup": "OK",
  "State": "Enabled"
},
"VLANs": {
  "@odata.id": "/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0/VLANs"
}
}

```

2. Get the BlueField's high-speed port's MAC addresses using the following Redfish command over the 1GbE OOB port to the BlueField BMC:

```

curl -k -u root:<password> -H "Content-Type: application/octet-stream" -X GET
https://<bmc_ip>/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/NetworkDevice
{
  "@odata.id":
"/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/NetworkDeviceFunctions/eth
  "@odata.type": "#NetworkDeviceFunction.v1_9_0.NetworkDeviceFunction",
  "Ethernet": {
    "MACAddress": "02:b1:b6:12:39:05",
    "MTUSize": 1500
  },
  "Id": "eth0f0",

```

```

"Links": {
  "OffloadSystem": {
    "@odata.id": "/redfish/v1/Systems/Bluefield"
  },
  "PhysicalPortAssignment": {
    "@odata.id":
"/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/Ports/eth0"
  }
},
"Name": "NetworkDeviceFunction",
"NetDevFuncCapabilities": [
  "Ethernet"
],
"NetDevFuncType": "Ethernet"
}

```

Step 8 – Change Mode of Operation to Zero-trust Mode

Unless it is explicitly desired for the host to be trusted, make sure to disable the host PCIe RShim to protect the BlueField from potential security threats from the host:

1. Use Redfish BIOS settings schema over the 1GbE OOB to the BlueField BMC:

```

curl -k -X PATCH -d '{"Attributes":{"Internal CPU Model": "Restricted"}}' -u root:<password>
https://<BF-BMC-IP>/redfish/v1/Systems/<SystemID>/Bios/Settings | python3 -m json.tool

```

The available BlueField host privilege levels are Restricted and Privileged. The default is Privileged, where the host has access to BlueField.

2. Change the privilege level to Restricted.

Note

Changing host privilege level requires BlueField reset for the change to take effect.

Info

For more information on BlueField modes of operation, refer to [this page](#).

Step 9 – (Optional) Change Mode of Operation from DPU Mode to NIC Mode

To change from [DPU mode to NIC mode](#) (or vice versa):

1. To enable NIC mode:

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

2. To disable NIC mode:

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

3. To check that the BMC recorded the change for the next UEFI reboot to apply it:

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

Note

Reset the BlueField (Arm and NIC) for the mode change to take effect.

4. To verify that the NIC mode has updated accordingly:

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

Step 10 – (Optional) Disable Secure Boot

As part of the default settings of the BlueField, UEFI Secure Boot is enabled and requires no special configuration to use it with the bundled Ubuntu OS shipped with the BlueField device. Disabling UEFI Secure Boot may be necessary when running an unsigned Arm OS image, such as a customer OS. Using Redfish Secure Boot schema over 1GbE to BlueField BMC, run:

```
curl -k -u root:<password> -H "Content-Type: application/octet-stream" -X GET https://<BF-BMC-IP>/redfish/v1/Systems/Bluefield/SecureBoot
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/SecureBoot",
  "@odata.type": "#SecureBoot.v1_1_0.SecureBoot",
  "Description": "The UEFI Secure Boot associated with this system.",
  "Id": "SecureBoot",
  "Name": "UEFI Secure Boot",
  "SecureBootCurrentBoot": "Enabled",
  "SecureBootEnable": true,
  "SecureBootMode": "SetupMode"
}
curl -k -u root:<BF-BMC-PASSWORD> -X PATCH https://<BF-BMC-IP>/redfish/v1/Systems/Bluefield/SecureBoot -H 'Content-Type: application/json' -d '{"SecureBootEnable": false}'
```

After running this command, the BlueField Arm OS must be rebooted twice. The first reboot is for the UEFI redfish client to read the request from the BMC and apply it; the second reboot is for the setting to take effect.

- From the BlueField BMC using Redfish:

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

- From RShim:

```
echo 'SW_RESET 1' > /dev/rshim0/misc
```

- From the BlueField Arm OS:

```
reboot
```

For more information on user management, review [this page](#).

NIC Mode Installation

The following sections detail the procedure for installing BlueField software when the BlueField networking platform (DPU or SuperNIC) is running in NIC mode.

Note

NIC mode is the default mode for BlueField SuperNICs, while BlueField DPUs are shipped with DPU mode as their default. To switch between the modes, see [NVIDIA BlueField Modes of Operation](#). To check which mode your BlueField is currently running:

- Using RedFish on the 1GbE interface, refer to [NVIDIA BlueField BMC Software Documentation](#)
- From the host server, refer to "[Enabling NIC Mode on BlueField-3 from Linux](#)"

- From the host server's UEFI BIOS menu, refer to "[Configuring NIC Mode on BlueField-3 from Host BIOS HII UEFI Menu](#)"

Note

In the out-of-box state of the BlueField the host is assumed to be trusted. Later in this procedure, after performing BFB Bundle update, a [step](#) is provided to disable the host RShim which the user may perform to protect the BlueField from potential security threats from the host.

Upgrade BlueField Firmware Components and BSP Using BFB Image

Upgrade the BlueField firmware components (i.e., ATF, UEFI, NIC-firmware, BlueField BMC firmware) and the BSP using the BFB image.

Tip

Make sure to download the latest bf-fwbundle image (BFB file) available from the [DOCA-Host and BlueField Bundle Runtime Downloads](#).

This can be performed using one of the following methods:

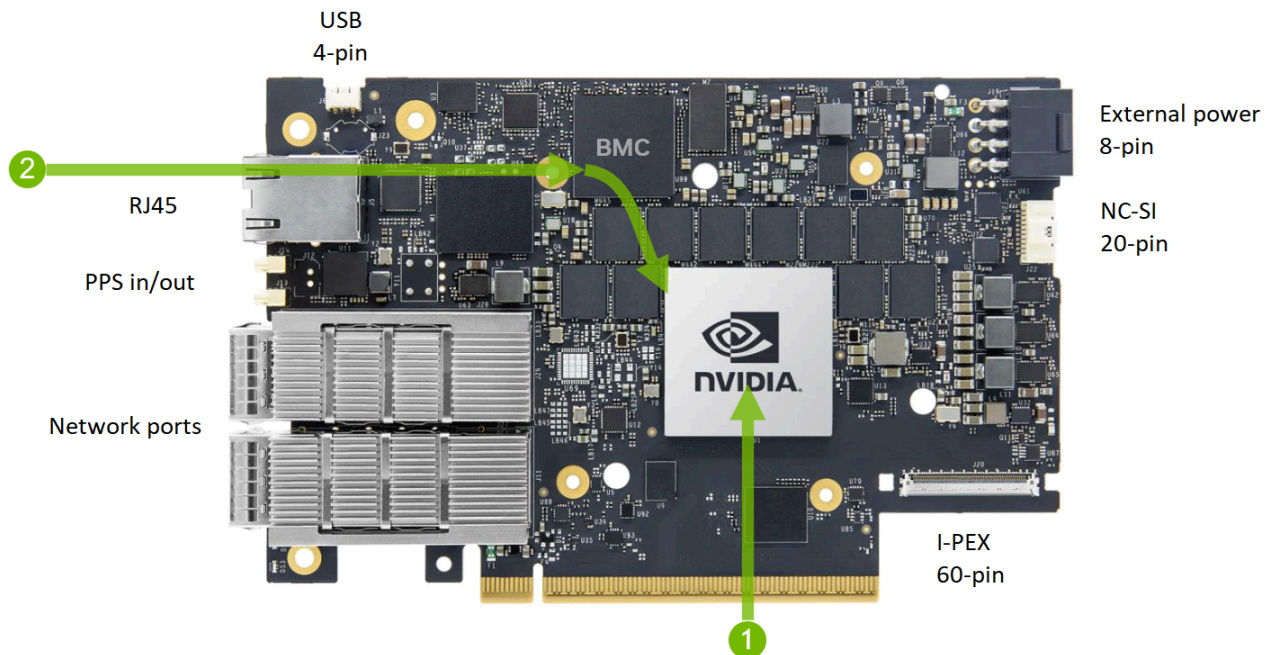
1. From the host x86, which should be considered as trusted during this maintenance window, follow the installation procedure [here](#).
2. If a DPU BMC connected to the ToR switch over 1GbE is available, follow the installation procedure [here](#).

PCIe (in-band, via host OS)

- 1 RShim over PCIe, BFB upgrade

RJ45 (out-of-band)

- 2 BlueField-BMC, RedFish, BFB upgrade



Changing UEFI and BMC 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.
- 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.

Change Mode of Operation to Zero-trust Mode

Unless it is explicitly desired for the host to be trusted, make sure to disable the host RShim to protect the BlueField from potential security threats from the host by running the following NC-SI command from the host BMC:

Set RShim State Command Format

Byte/Bit	31:24	23:16	15:8	7:0
0...15	NC-SI Header (OEM Command)			
16:19	NVIDIA Manufacture ID (IANA) = 0x8119			
20:23	Command rev=0x00	MLNX Cmd ID= 0x12	Parameter=0x1B	Reserved
24:27	Reserved			Host_RT_Access_State
28:31	Checksum 31:0			

Set RShim State Command Parameters

Field	Bytes	Offset in NC-SI Command	Description
Host_RT_Access_State	1	27	RShim state: <ul style="list-style-type: none"> • 0 – Enabled • 1 – Locked • Other – reserved

Day 2 Management Operations

After provisioning and deploying the BlueField for the first time, there are several activities required from time-to-time throughout the BlueField's lifecycle.

Info

Before proceeding, review section "[Management Methods](#)" to learn about the recommended management methods for specific tasks on the BlueField device.

Modular Update

This upgrade procedure enables upgrading DOCA components using standard Linux tools (e.g., apt update and yum update). This process utilizes native package manager repositories to upgrade BlueField networking platforms (DPUs or SuperNICs) without the need for a full installation.

This process has the following benefits :

- Only updates components that include modifications
 - Configurable – user can select specific components (e.g., UEFI-ATF, NIC-FW)
- Includes upgrade of:
 - DOCA drivers and libraries
 - DOCA reference applications

- BSP (UEFI/ATF) upgrade while maintaining the configuration
- NIC firmware upgrade while maintaining the configuration
- Does not:
 - Impact user binaries
 - Upgrade non-Ubuntu OS kernels
 - Upgrade BlueField BMC firmware
- After completion of BlueField upgrade:
 - If NIC firmware was not updated, perform BlueField Arm reset (software reset/reboot BlueField)
 - If NIC firmware was updated, perform firmware reset (mlxfwreset) or perform a graceful shutdown and power cycle

OS	Action	Instructions
Ubuntu/ Debian	Remove mlxbf-bootimages package	<pre><bf> \$ apt remove --purge mlxbf-bootimages* -y</pre>
	Install the the GPG key	<pre><bf> \$ apt update <bf> \$ apt install gnupg2</pre>
	Export the desired distribution	<p>Export DOCA_REPO with the relevant URL. The following is an example for Ubuntu 22.04:</p> <pre><bf> \$ export DOCA_REPO="https://linux.mellanox.com/public/repo/doca/2.7.0/ubuntu22.04/dpu-arm64"</pre> <ul style="list-style-type: none"> ● Ubuntu 22.04 – https://linux.mellanox.com/public/repo/doca/2.7.0/ubuntu22.04/dpu-arm64

OS	Action	Instructions
		<ul style="list-style-type: none"> • Ubuntu 20.04 – https://linux.mellanox.com/public/repo/docca/2.7.0/ubuntu20.04/dpu-arm64 • Debian 12 – https://linux.mellanox.com/public/repo/docca/2.7.0/debian12/dpu-arm64
	Add GPG key to APT trusted keyring	<pre><bf> \$ curl \$DOCA_REPO/GPG-KEY-Mellanox.pub gpg --dearmor > /etc/apt/trusted.gpg.d/GPG-KEY-Mellanox.pub</pre>
	Add DOCA online repository	<pre><bf> \$ echo "deb [signed-by=/etc/apt/trusted.gpg.d/GPG-KEY-Mellanox.pub] \$DOCA_REPO ./" > /etc/apt/sources.list.d/docca.list</pre>
	Update index	<pre><bf> \$ apt update</pre>
	Upgrade UEFI/ATF firmware	<p>Run:</p> <pre><bf> \$ apt install mlxbf-bootimages-signed</pre> <p>Then i nitiate upgrade for UEFI/ATF firmware:</p> <pre><bf> \$ apt install mlxbf-scripts <bf> \$ bfrec</pre>
	Upgrade BlueField NIC firmware	<p>Run:</p> <pre><bf> \$ apt install mlnx-fw-updater-signed.aarch64</pre> <div style="background-color: #ffffcc; padding: 10px; margin-top: 10px;"> <p>Note This immediately starts NIC firmware upgrade.</p> </div>

OS	Action	Instructions
		<p>To prevent automatic upgrade, run:</p> <pre><bf> \$ export RUN_FW_UPDATER=no</pre>
	Remove old metapackages	<pre><bf> \$ apt-get remove doca-tools doca-sdk doca-runtime -y</pre>
	Install new metapackages	<pre><bf> \$ apt-get install doca-runtime doca-devel -y</pre>
	Upgrade system	<pre><bf> \$ apt upgrade</pre>
	Apply the new changes, NIC firmware, and UEFI/ATF	<p>For the upgrade to take effect, perform BlueField system reboot as explained in the "NVIDIA BlueField Reset and Reboot Procedures" troubleshooting page.</p> <div style="background-color: #ffffcc; padding: 10px;"> <p>i Note This step triggers immediate reboot of the BlueField Arm cores.</p> </div>
Cent OS/RHEL/Anolis/Rocky	Remove mlxbf-bootimages package	<pre><bf> \$ yum -y remove mlxbf-bootimages* <bf> \$ yum makecache</pre>
	Export the desired distribution	<p>Export DOCA_REPO with the relevant URL. The following is an example for Rocky Linux 8.6:</p> <pre><bf> \$ export DOCA_REPO="https://linux.mellanox.com/public/repo/doca/2.7.0/rhel8.6/dpu-arm64/"</pre>

OS	Action	Instructions
		<ul style="list-style-type: none"> • AnolisOS 8.6 – https://linux.mellanox.com/public/repo/doca/2.7.0/anolis8.6/dpu-arm64/ • OpenEuler 20.03 sp1 – https://linux.mellanox.com/public/repo/doca/2.7.0/openeuler20.03sp1/dpu-arm64/ • CentOS 7.6 with 4.19 kernel – https://linux.mellanox.com/public/repo/doca/2.7.0/rhel7.6-4.19/dpu-arm64/ • CentOS 7.6 with 5.10 kernel – https://linux.mellanox.com/public/repo/doca/2.7.0/rhel7.6-5.10/dpu-arm64/ • CentOS 7.6 with 5.4 kernel – https://linux.mellanox.com/public/repo/doca/2.7.0/rhel7.6/dpu-arm64/ • Rocky Linux 8.6 – https://linux.mellanox.com/public/repo/doca/2.7.0/rhel8.6/dpu-arm64/
	Add DOCA online repository	<pre>echo "[doca] name=DOCA Online Repo baseurl=\$DOCA_REPO enabled=1 gpgcheck=0 priority=10 cost=10" > /etc/yum.repos.d/doca.repo</pre> <p>A file is created under /etc/yum.repos.d/doca.repo .</p>
	Update index	<pre><bf> \$ yum makecache</pre>
	Upgrade UEFI/ATF firmware	<p>Run:</p> <pre><bf> \$ yum install mlxbf-bootimages-signed.aarch64 mlxbf-bfscripts</pre> <p>Then i nitiate the upgrade for UEFI/ATF firmware:</p>

OS	Action	Instructions
		<pre data-bbox="527 268 675 296"><bf> \$ bfrec</pre>
	Upgrade BlueField NIC firmware	<p data-bbox="469 380 1349 457">The following command updates the firmware package and automatically attempts to flash the firmware to the NIC:</p> <pre data-bbox="527 514 1143 541"><bf> \$ yum install mlnx-fw-updater-signed.aarch64</pre> <div data-bbox="500 695 1240 873" style="background-color: #ffffcc; padding: 10px;"> <p>i Info This step can be used as a standalone firmware update. In any case, it is performed as part of the upgrade flow.</p> </div> <div data-bbox="500 1024 1256 1161" style="background-color: #ffffcc; padding: 10px;"> <p>i Note To prevent automatic flashing of the firmware to the NIC, run the following first:</p> </div> <pre data-bbox="576 1213 1179 1339"><bf> \$ export RUN_FW_UPDATER=no 00000191-d778-dc14-afb7-dff99aa90006</pre>
	Remove old metapackages	<pre data-bbox="527 1528 1208 1556"><bf> \$ yum -y remove doca-tools doca-sdk doca-runtime</pre>
	Install new metapackages	<pre data-bbox="527 1686 1078 1713"><bf> \$ yum -y install doca-runtime doca-devel</pre>
	Upgrade system	<pre data-bbox="527 1843 878 1871"><bf> \$ yum upgrade --nobest</pre>

OS	Action	Instructions
	Apply the new changes, NIC firmware, and UEFI/ATF	<p>For the upgrade to take effect, perform BlueField system reboot as explained in the "NVIDIA BlueField Reset and Reboot Procedures" troubleshooting page.</p> <div style="background-color: #ffffcc; padding: 10px; border: 1px solid #ccc;"> <p>Note This step triggers immediate reboot of the BlueField Arm cores.</p> </div>

Resetting BlueField to Factory Defaults

Users may want to reset the BlueField to factory defaults. To do that, it is necessary to reset to default the BlueField BMC, BlueField UEFI, NIC, and the Arm. Follow the steps in the subsections below for more.

Step 1 – Reset BlueField BMC to Factory Default

1. Run the following command:

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST https://<BF-BMC-IP>/redfish/v1/Managers/Bluefield_BMC/Actions/Manager.ResetToDefaults -d '{"ResetToDefaultsType": "ResetAll"}'
```

```
{
  "@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The request completed successfully.",
      "MessageArgs": [],
      "MessageId": "Base.1.15.0.Success",
      "MessageSeverity": "OK",
      "Resolution": "None"
    }
  ]
}
```

2. Reboot the BMC for the factory reset to take effect:

```
> curl -k -u root:<password> -H "Content-Type: application/json" -X POST -d '{"ResetType":
"GracefulRestart"}' https://<BF-BMC-
IP>/redfish/v1/Managers/Bluefield_BMC/Actions/Manager.Reset
{
  "@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The request completed successfully.",
      "MessageArgs": [],
      "MessageId": "Base.1.13.0.Success",
      "MessageSeverity": "OK",
      "Resolution": "None"
    }
  ]
}
```

Step 2 – Wipe BlueField eMMC and SSD Storage

Users may wipe their eMMC and NVMe storage using the following Redfish commands:

- eMMC:

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

- NVMe:

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

Step 3 – Reset UEFI to Factory Default

Use the Redfish BIOS Settings PATCH command:

```
curl -k -u root:'<password>' -X PATCH -d '{"Attributes":{"ResetEfiVars": true}}' https://<BF-BMC-IP>/redfish/v1/Systems/<SystemID>/Bios/Settings | python3 -m json.tool
```

Step 4 – Reset NIC TLVs to Factory Default

Run the following from the Arm console:

```
bf> mlxconfig -d <device> -y reset
```

Appendixes

The following appendixes are available:

- [Appendix - BMC and eROT Upgrade Process for BlueField-2](#)
- [Appendix - BlueField Management in DPU Mode](#)
- [Appendix - BlueField Management in NIC Mode](#)
- [Appendix - BlueField DHCP Discover](#)
- [Appendix - Using Boot Order Schema to Set Boot Order](#)
- [Appendix - Relating BlueField to Host and Port](#)

Appendix - BMC and eROT Upgrade Process for BlueField-2

To upgrade BlueField BMC firmware on a BlueField-2 device, follow this procedure:

1. Download openbmctool v1.22 and resolve dependencies.
 - `sudo pip3 install paramiko`
 - `sudo pip3 install scp`
2. Trigger BMC update.

```
python3 openbmctool.py -H <BF-BMC-IP> -U root -P <password> firmware flash bmc -f <path-to-signed-bmc-image>
```

After initiating the BMC firmware secure update, a new task is created. Example:


```
Attempting login...
Uploading file to BMC
Upload complete.
Firmware activation is in progress. Please wait for activation task id="0" to get completed before
rebooting the bmc.
User root has been logged out
```

Info

BMC firmware update takes 15-20 mins .

3. Track the progress of the update by using the task id received in the response above (i.e., 0) in your query and monitoring the value of the task's TaskProgress field.

```
python3 openbmctool.py -H <BF-BMC-IP> -U root -P <password> task status -i <task_Id>
```

Before proceeding to other operations. Keep running this command until it outputs:

```
TaskState="Completed"
TaskStatus="OK"
TaskProgress="100"
```

4. Reset/reboot the BMC.

Info

This step may be skipped if you intend to perform eROT update.

```
python3 openbmctool.py -H <BF-BMC-IP> -U root -P <password> firmware running_version
```

Info

Wait a few seconds before attempting to log back into the BMC as it loses connection during and shortly after reboot.

5. Trigger eROT update.

```
python3 openbmctool.py -H <ip_address> -U root -P <BF-BMC-IP> apfirmware flash cec -f <path-to-signed-CEC-OTA-image-file>
```

After initiating the eROT update, the following indication is provided:

```
Uploading firmware image: 100.00%  
Firmware update for cec triggered successfully.
```

6. Perform host power cycle for the eROT update to take effect:

```
sudo ipmitool power cycle
```

Appendix - BlueField Management in DPU Mode

Management in BlueField BMC

The BlueField networking platform (DPU or SuperNIC) incorporates an integrated BMC, ASPEED AST2600. The on-board BMC provides security in untrusted platforms and is therefore needed in most BlueField use cases.

Like the host BMC, managed host platform, the BlueField BMC is a trusted entity (with its own ERoT to ensure that its firmware is secured) that enables provisioning and managing the BlueField over a separated management network, using standard interfaces, protocols, and security to manage the full lifecycle of the BlueField. In addition, the BlueField BMC enables managing the BlueField even if the BlueField's OS is down, and it has a separate power input so it can also hard reset the BlueField if required.

The main interface for the BlueField BMC is a 1GbE RJ45 OOB management port that is connected to the internal management Ethernet network of the cloud service provider or the Enterprise IT management network.

Managing the BlueField using its BMC is detailed hereafter.

Remote Management Using Redfish Protocol

Supported by BlueField, the Redfish standard is a suite of specifications that delivers an industry standard protocol providing a secured RESTful interface for the management of servers, storage, networking, and converged infrastructure.

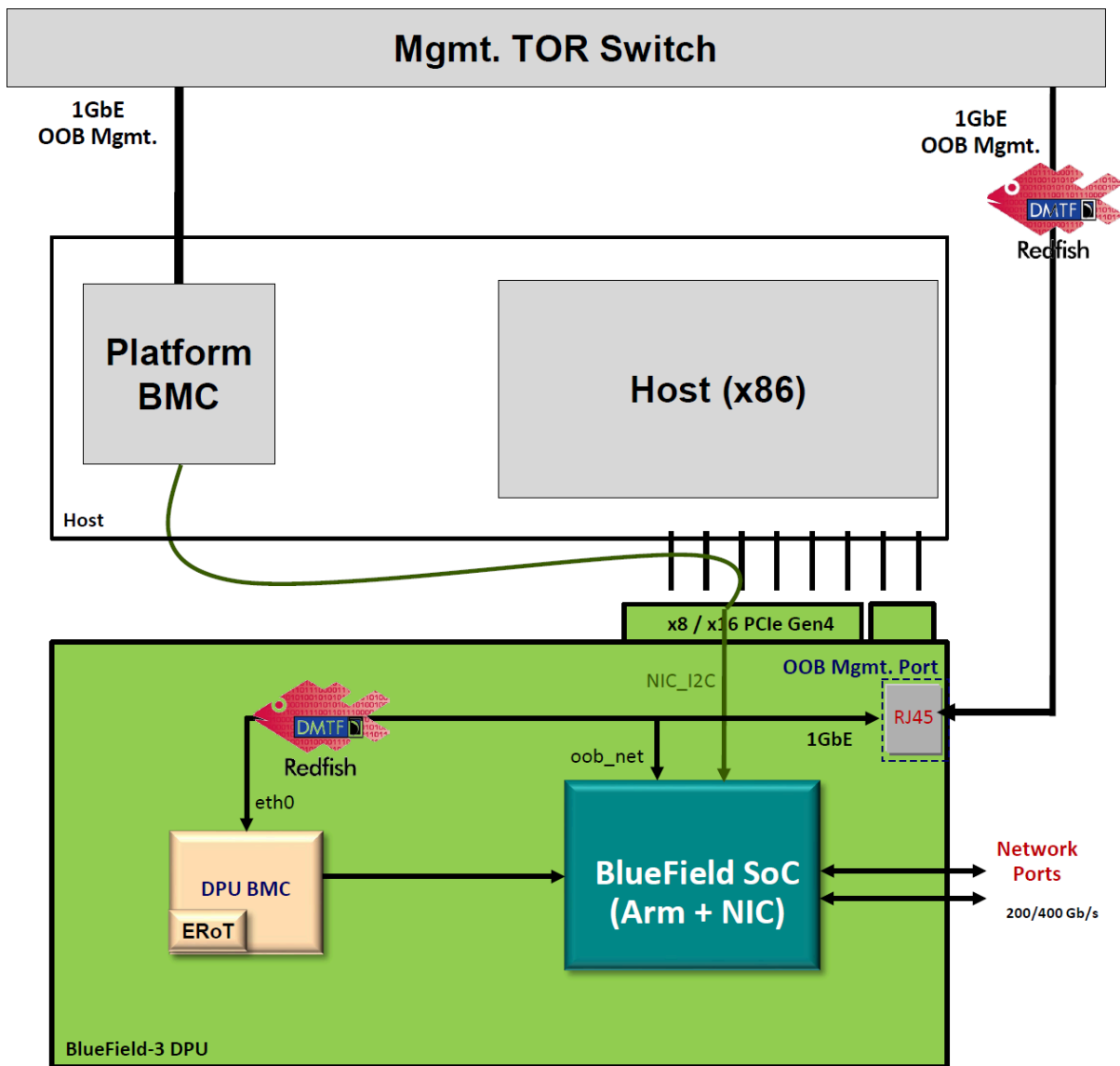
Redfish is supported from the host BMC and BlueField BMC.

Redfish replaces IPMI, providing the following advantages:

- Human readable schemas
- Interoperable, equally usable by apps, GUIs, and scripts
- Extensible to add capabilities
- Secured using HTTPS

Management Architecture

The following diagram illustrates the architecture and connectivity for managing the BlueField.



Management Interfaces

i Info


See [this page](#) for a detailed description of the interfaces of BlueField-3.

i Info

See [here](#) page for a detailed description of the interfaces of BlueField-2.

The following table describes the interfaces available to manage the BlueField.

Management Interface	Description	Comment
OOB Management Port (1GbE RJ45)	A dedicated, separate Ethernet interface to manage the BlueField from the remote management controller (RMC)	<div data-bbox="630 737 1463 1010" style="background-color: #ffffcc; padding: 10px;"> <p>Info NVIDIA recommends using this interface as the main management interface.</p> </div> <p>Enables managing the BlueField life cycle using the BlueField's BMC. Supports Redfish commands to the BlueField BMC (eth0). Recovery flows, monitoring, and configuration operations are all available through this interface. In addition, this physical interface allows users to SSH directly to the BlueField (oob_net).</p> <div data-bbox="630 1436 1463 1751" style="background-color: #ffffcc; padding: 10px;"> <p>Note IPMI is supported for backward compatibility, but it is recommended to start new deployments with Redfish only.</p> </div>
SMBus (PCIe)	Enables PLDM/NC-SI over MCTP between	Enables the host BMC to monitor the BlueField

Management Interface	Description	Comment
Golden Fingers)	the BlueField and the host BMC	
PCIe	PCIe interface between the BlueField and host server	<p>Enables the host to recover the BlueField using RShim PCIe physical function (PF) when the host is trusted</p> <div data-bbox="630 646 1463 919" style="background-color: #ffffcc; padding: 10px;"> <p> Info Unavailable while in zero-trust mode. Use the 1GbE OOB interface instead.</p> </div>

Recommended Management Approach

The BlueField BMC allows managing the BlueField over the 1GbE OOB interface using Redfish protocol. The following functions are available:

- BlueField and BlueField BMC upgrade and recovery
- Monitoring of the BlueField device
- BlueField BlueField reset control (even when BlueField OS is halted)
- Setting BlueField UEFI configuration
- Console interface to BlueField

Management Methods

The following subsections describe the recommended management methods for specific tasks on the BlueField .

Note

The following method are not supported by BlueField SuperNIC.

BlueField Update and Recovery

The NVIDIA BlueField offers two file format for performing software upgrade:

- The standard ISO format
- BlueField bootstream (BFB) file format.

Each file format has a different update and recovery method:

- For ISO, a PXE server may be used to load an ISO image which contains the necessary updates. This can be accomplished using BlueField's UEFI, interface PXE server over the 1GbE OOB or through the high-speed data ports. BlueField SoC runs UEFI/PXE which sends a DHCP DISCOVER over the 1GbE OOB interface, including vendor class ("NVIDIA/BF/PXE") for BlueField SoC (to allow customer's server to differentiate between BlueField SoC and BlueField BMC), and MAC for identification and discovery.
- BFB serves both as a comprehensive upgrade tool and a recovery solution for the BlueField. There are two ways to facilitate these upgrade and recovery methods:
 - The BlueField BMC is under the control of a remote management controller (RMC) that utilizes the Redfish protocol over the 1GbE OOB connection. A pre-installed golden image can be used which allows flash and recovery of the BlueField devices. This can be triggered by either the RMC or a trusted platform's BMC. For more information, refer to [this](#) page
 - The host BMC is under the control of a RMC that utilized Redfish protocol over PCIe connection.

(i) Note

After an upgrade/recovery, a system power cycle may be required to apply changes.

BlueField BMC Update

The BlueField BMC can be updated by the RMC using Redfish over the 1GbE OOB port to the BlueField BMC.

BlueField BMC update is A/B redundant, using a dual firmware flash. If both flashes fail to boot, the BlueField BMC may be recovered from the platform's BMC using the SMBus or UART interfaces.

Please refer to [this](#) page for more information.

(i) Note

After an upgrade, a system power cycle may be required to apply changes.

BlueField Monitoring and Telemetry

The RMC may monitor and read telemetry of the BlueField using Redfish over the 1GbE OOB port to the BlueField BMC.

- BlueField temperatures (board, DDR, and ports), voltages and link states
- BlueField FRU information about NIC FW, CPU, DDR, eMMC, network interface, etc.

- Device sensor data record (SDR), sensor threshold and events, system event logs (SEL), etc.

Please refer to [this](#) page for more information.

BlueField and BlueField BMC Reset Control

The RMC may issue a reset to the BlueField (soft or hard) or to the BlueField BMC, both using Redfish over the 1GbE OOB port to the BlueField BMC.

Please refer to these pages ([1,2](#)) for more information.

BlueField UEFI Configuration

BlueField UEFI settings may be modified using Redfish over the 1GbE OOB port to the BlueField BMC. This includes changing UEFI default password (which is mandatory), setting BlueField to zero-trust, setting date and time, etc.

Please refer to [this](#) page for more information.

Console Interface

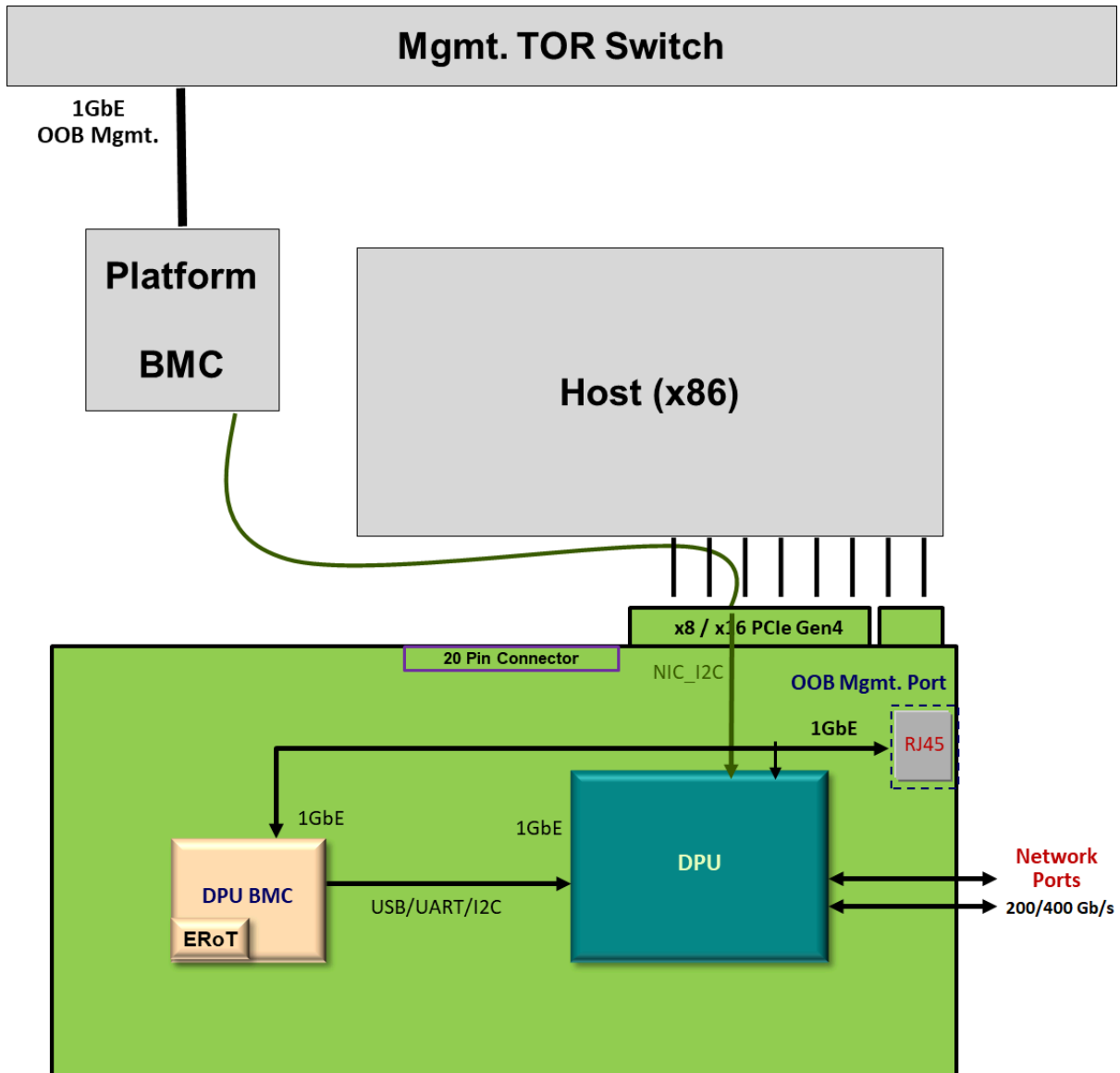
The BlueField console interface is accessible via the BlueField BMC using Serial-over-LAN (SoL) over the 1GbE OOB port. The RMC may access the console interface of the BlueField device to track its boot progress.

Please refer to [this](#) page for more information.

Appendix - BlueField Management in NIC Mode

Management Architecture

The following diagram illustrates the architecture and connectivity for managing the BlueField in NIC mode.



Management Interfaces

i Info

See [this](#) page for a detailed description of the interfaces of BlueField-3.

i Info

See [this](#) page for a detailed description of the interfaces of BlueField-2.

The following table describes the interfaces available to manage the BlueField.

Management Interface	Description	Comment
SMBus (PCIe Golden Fingers)	Enables PLDM/NC-SI over MCTP between the BlueField and the host BMC	Enables the host BMC to monitor and reset the BlueField
PCIe	PCIe interface between the BlueField and host x86 server	Enables the host x86 to update and recover the BlueField using RShim PCIe physical function (PF) when the host is trusted i Info Unavailable while in zero-trust mode. Use the 1GbE OOB interface instead.
20-pin connector UART/USB/I2C with	Not in use in the recommended approach	

Management Interface	Description	Comment
DPU BMC Data ports		

Recommended Management Approach

The following are recommended approaches for BlueField management in NIC mode:

- From platform BMC using NCSI over SMBus. This method enables the following capabilities:
 - Monitoring of the BlueField device
 - Reset control of the BlueField device
- From host x86 during a maintenance window. This method enables the following capabilities:
 - BlueField and upgrade and recovery

Note

It is required to enable host PCIe RShim for this maintenance timeslot for upgrade

Management Methods

The following subsections describe the recommended management methods for specific tasks on the BlueField .

BlueField Update and Recovery

The NVIDIA BlueField file format for performing software upgrade is BlueField bootstream (BFB). The BFB serves both as a comprehensive upgrade and recovery solution for the BlueField.

Host PCIe RShim is used to push the BFB to facilitate the upgrade and recovery. Please refer to [this page](#) for more information.

Note

After an upgrade/recovery, a system power cycle may be required to apply changes.

BlueField Monitoring

BlueField monitoring may be performed from the platform BMC, using NC-SI over SMBus to obtain the following information:

- BlueField temperatures (board, DDR, and ports), voltages, and link states
- BlueField FRU information about NIC firmware, network interfaces, etc.
- Device sensor data record (SDR), sensor threshold and events, system event logs (SEL), etc.

Request the *NVIDIA Specific NC-SI OEM Commands Application Note* from your NVIDIA representative for more details.

BlueField and Reset Control

The platform BMC may issue a (soft or hard) reset to the BlueField using NC-SI over SMBus.

Request the *NVIDIA Specific NC-SI OEM Commands Application Note* from your NVIDIA representative for more details.

Appendix - BlueField DHCP Discover

Possible Vendor Classes

Vendor Class String in DHCP Discover	Originator of the DHCP Discover
NVIDIA/BF/OOB	BlueField SoC Arm OS over 1GbE OOB
NVIDIA/BF/PXE	BlueField SoC UEFI BIOS over either 1GbE OOB or high-speed ports
NVIDIA/BF/DP	BlueField SoC Arm OS over high-speed ports
NVIDIA/BF/BMC	BlueField BMC over 1GbE OOB
BF2Client	BlueField OS/UEFI in older versions
PXEClient	BlueField OS/UEFI in older versions
HTTPClient	BlueField OS/UEFI in older versions

BlueField DHCP DISCOVER

Sent from the BlueField UEFI BIOS to the DHCP server:

```
0.0.0.0 -> 255.255.255.255 DHCP 370 DHCP Discover - Transaction ID 0xfd2f32f1
192.168.254.119 -> 255.255.255.255 DHCP 342 DHCP Offer - Transaction ID 0xfd2f32f1
0.0.0.0 -> 255.255.255.255 DHCP 382 DHCP Request - Transaction ID 0xfd2f32f1
192.168.254.119 -> 255.255.255.255 DHCP 342 DHCP ACK - Transaction ID 0xfd2f32f1
aa:bb:cc:dd:ee:ff -> Broadcast ARP 60 Who has 192.168.254.119? Tell 192.168.254.139
gg:hh:ii:jj:kk:ll -> aa:bb:cc:dd:ee:ff ARP 60 192.168.254.119 is at gg:hh:ii:jj:kk:ll
192.168.254.139 -> 192.168.254.119 TFTP 98 Read Request, File: /grubaa64.efi, Transfer type: octet,
tsize\000=0\000, blksize\000=1400\000, windowsize\000=4\000
192.168.254.119 -> 192.168.254.139 TFTP 84 Option Acknowledgement, tsize\000=2414472\000,
blksize\000=1400\000, windowsize\000=4\000
```

192.168.254.139 -> 192.168.254.119 TFTP 72 Error Code, Code: Option negotiation failed, Message: User aborted the transfer

192.168.254.139 -> 192.168.254.119 TFTP 90 Read Request, File: /grubaa64.efi, Transfer type: octet, blksize\000=1400\000, windowsize\000=4\000

192.168.254.119 -> 192.168.254.139 TFTP 70 Option Acknowledgement, blksize\000=1400\000, windowsize\000=4\000

192.168.254.139 -> 192.168.254.119 TFTP 60 Acknowledgement, Block: 0

192.168.254.119 -> 192.168.254.139 TFTP 1446 Data Packet, Block: 1

192.168.254.119 -> 192.168.254.139 TFTP 1446 Data Packet, Block: 2

192.168.254.119 -> 192.168.254.139 TFTP 1446 Data Packet, Block: 3

Ethernet II, Src: aa:bb:cc:dd:ee:ff (aa:bb:cc:dd:ee:ff), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

Destination: Broadcast (ff:ff:ff:ff:ff:ff)

Address: Broadcast (ff:ff:ff:ff:ff:ff)

.... ..1. = LG bit: Locally administered address (this is NOT the factory default)

.... ..1 = IG bit: Group address (multicast/broadcast)

Source: aa:bb:cc:dd:ee:ff (aa:bb:cc:dd:ee:ff)

Address: aa:bb:cc:dd:ee:ff (aa:bb:cc:dd:ee:ff)

.... ..0. = LG bit: Globally unique address (factory default)

.... ..0 = IG bit: Individual address (unicast)

Type: IP (0x0800)

Internet Protocol Version 4, Src: 0.0.0.0 (0.0.0.0), Dst: 255.255.255.255 (255.255.255.255)

Version: 4

Header length: 20 bytes

Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))

0000 00.. = Differentiated Services Codepoint: Default (0x00)

.... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport) (0x00)

Total Length: 368

Identification: 0x61c7 (25031)

Flags: 0x00

0... = Reserved bit: Not set

.0.. = Don't fragment: Not set

..0. = More fragments: Not set

Fragment offset: 0

Time to live: 64

Protocol: UDP (17)

Header checksum: 0x17b7 [validation disabled]

[Good: False]

[Bad: False]

Source: 0.0.0.0 (0.0.0.0)

Destination: 255.255.255.255 (255.255.255.255)

User Datagram Protocol, Src Port: bootpc (68), Dst Port: bootps (67)

Source port: bootpc (68)

Destination port: bootps (67)
Length: 348
Checksum: 0xe557 [validation disabled]
[Good Checksum: False]
[Bad Checksum: False]

Bootstrap Protocol

Message type: Boot Request (1)
Hardware type: Ethernet (0x01)
Hardware address length: 6
Hops: 0
Transaction ID: 0xfd2f32f1
Seconds elapsed: 0
Bootp flags: 0x8000 (Broadcast)
1... .. = Broadcast flag: Broadcast
.000 0000 0000 0000 = Reserved flags: 0x0000
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: aa:bb:cc:dd:ee:ff (aa:bb:cc:dd:ee:ff)
Client hardware address padding: 00000000000000000000
Server host name not given
Boot file name not given
Magic cookie: DHCP
Option: (53) DHCP Message Type
Length: 1
DHCP: Request (3)
Option: (54) DHCP Server Identifier
Length: 4
DHCP Server Identifier: 192.168.254.119 (192.168.254.119)
Option: (50) Requested IP Address
Length: 4
Requested IP Address: 192.168.254.139 (192.168.254.139)
Option: (57) Maximum DHCP Message Size
Length: 2
Maximum DHCP Message Size: 65280
Option: (55) Parameter Request List
Length: 35
Parameter Request List Item: (1) Subnet Mask
Parameter Request List Item: (2) Time Offset
Parameter Request List Item: (3) Router
Parameter Request List Item: (4) Time Server
Parameter Request List Item: (5) Name Server
Parameter Request List Item: (6) Domain Name Server

Parameter Request List Item: (12) Host Name
Parameter Request List Item: (13) Boot File Size
Parameter Request List Item: (15) Domain Name
Parameter Request List Item: (17) Root Path
Parameter Request List Item: (18) Extensions Path
Parameter Request List Item: (22) Maximum Datagram Reassembly Size
Parameter Request List Item: (23) Default IP Time-to-Live
Parameter Request List Item: (28) Broadcast Address
Parameter Request List Item: (40) Network Information Service Domain
Parameter Request List Item: (41) Network Information Service Servers
Parameter Request List Item: (42) Network Time Protocol Servers
Parameter Request List Item: (43) Vendor-Specific Information
Parameter Request List Item: (50) Requested IP Address
Parameter Request List Item: (51) IP Address Lease Time
Parameter Request List Item: (54) DHCP Server Identifier
Parameter Request List Item: (58) Renewal Time Value
Parameter Request List Item: (59) Rebinding Time Value
Parameter Request List Item: (60) Vendor class identifier
Parameter Request List Item: (66) TFTP Server Name
Parameter Request List Item: (67) Bootfile name
Parameter Request List Item: (97) UUID/GUID-based Client Identifier
Parameter Request List Item: (128) DOCSIS full security server IP [TODO]
Parameter Request List Item: (129) PXE - undefined (vendor specific)
Parameter Request List Item: (130) PXE - undefined (vendor specific)
Parameter Request List Item: (131) PXE - undefined (vendor specific)
Parameter Request List Item: (132) PXE - undefined (vendor specific)
Parameter Request List Item: (133) PXE - undefined (vendor specific)
Parameter Request List Item: (134) PXE - undefined (vendor specific)
Parameter Request List Item: (135) PXE - undefined (vendor specific)
Option: (97) UUID/GUID-based Client Identifier
Length: 17
Client Identifier (UUID): c24e2ad6-a730-ec11-8000-08c0eb58d8ec
Option: (94) Client Network Device Interface
Length: 3
Major Version: 3
Minor Version: 16
Option: (93) Client System Architecture
Length: 2
Client System Architecture: Unknown (11)
Option: (60) Vendor class identifier
Length: 13
Vendor class identifier: NVIDIA/BF/PXE
Option: (255) End
Option End: 255

Note

Note the "Vendor class identifier" value in line 134.

Appendix - Using Boot Order Schema to Set Boot Order

1. Check the current boot order by doing GET on the ComputerSystem schema over 1GbE OOB to the BlueField BMC. Look for the BootOrder attribute under the Boot.

```
curl -k -X GET -u root:<password> https://<BF-BMC-IP>/redfish/v1/Systems/<SystemID>/ |  
python3 -m json.tool  
{  
  ....  
  "Boot": {  
    ....  
    "BootOrder": [  
      "Boot0017",  
      "Boot0001",  
      "Boot0002",  
      "Boot0003",  
      "Boot0004",  
      "Boot0005",  
      "Boot0006",  
      "Boot0007",  
    ],  
    ....  
  }  
  ....  
}
```

2. To get the details of a particular entity in the BootOrder array, perform a GET to the respective BootOption URL over 1GbE OOB to the BlueField BMC. For example, to

get details of Boot0006, run:

```
curl -k -X GET -u root:<password> https://<BF-BMC-IP>/redfish/v1/Systems/<SystemID>/BootOptions/Boot0006 | python3 -m json.tool

{
  "@odata.type": "#BootOption.v1_0_3.BootOption",
  "@odata.id": "/redfish/v1/Systems/SystemId/BootOptions/Boot0006",
  "Id": "Boot0006",
  "BootOptionEnabled": true,
  "BootOptionReference": "Boot0006",
  "DisplayName": "UEFI HTTPv6 (MAC:B8CEF6B8A006)",
  "UefiDevicePath":
  "PciRoot(0x0)/Pci(0x0,0x0)/Pci(0x0,0x0)/Pci(0x0,0x0)/Pci(0x0,0x0)/MAC(B8CEF6B8A006,0x1)/IPV6(000
}
```

3. To change the boot order, the entire `BootOrder` array must be PATCHed to the pending settings URI. For this example of the `BootOrder` array, if you intend to have `Boot0006` at the beginning of the array, then the PATCH operation is as follows:

Note

Updating the `BootOrder` array results in a permanent boot order change (persistent across reboots).

```
curl -k -u root:<password> -X PATCH -d '{ "Boot": { "BootOrder": [ "Boot0006", "Boot0017", "Boot0001", "Boot0002", "Boot0003", "Boot0004", "Boot0005", "Boot0007", ] }}' https://<BF-BMC-IP>/redfish/v1/Systems/<SystemID>/Settings | python3 -m json.tool
```

4. After a successful PATCH, reboot the BlueField and check if the settings have been applied by doing a GET on the `ComputerSystem` schema.
5. If the `BootOrder` array is updated as intended then the settings have been applied and the BlueField should boot as per the order in preceding cycles.

Appendix - Relating BlueField to Host and Port

1. Get the BlueField's BMC MAC address using the following Redfish command over the 1GbE OOB port to the BlueField BMC:

```
curl -k -u root:<password> -H 'Content-Type: application/json' -X GET https://<BF-BMC-IP>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0
{
  "@odata.id": "/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0",
  "@odata.type": "#EthernetInterface.v1_6_0.EthernetInterface",
  "DHCPv4": {
    "DHCPEnabled": true,
    "UseDNSServers": true,
    "UseDomainName": true,
    "UseNTPServers": true
  },
  "DHCPv6": {
    "OperatingMode": "Stateful",
    "UseDNSServers": true,
    "UseDomainName": true,
    "UseNTPServers": true
  },
  "Description": "Management Network Interface",
  "FQDN": "dpu-bmc",
  "HostName": "dpu-bmc",
  "IPv4Addresses": [
    {
      "Address": "10.237.40.179",
      "AddressOrigin": "DHCP",
      "Gateway": "0.0.0.0",
      "SubnetMask": "255.255.0.0"
    }
  ],
  "IPv4StaticAddresses": [],
  "IPv6AddressPolicyTable": [],
  "IPv6Addresses": [
    {
      "Address": "fdfd:fdfd:10:237:966d:aeff:fe17:9f5f",
      "AddressOrigin": "DHCPv6",
```

```

    "AddressState": null,
    "PrefixLength": 64
  },
  {
    "Address": "fe80::966d:aeff:fe17:9f5f",
    "AddressOrigin": "LinkLocal",
    "AddressState": null,
    "PrefixLength": 64
  }
],
"IPv6DefaultGateway": "fe80::445b:ed80:5f97:8900",
"IPv6StaticAddresses": [],
"Id": "eth0",
"InterfaceEnabled": true,
"LinkStatus": "LinkUp",
"MACAddress": "94:6d:ae:17:9f:5f",
"MTUSize": 1500,
"Name": "Manager Ethernet Interface",
"NameServers": [
  "fdfd:fdfd:7:77:250:56ff:fe8b:e4f9"
],
"SpeedMbps": 0,
"StaticNameServers": [],
"Status": {
  "Health": "OK",
  "HealthRollup": "OK",
  "State": "Enabled"
},
"VLANs": {
  "@odata.id": "/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0/VLANs"
}
}

```

2. Get the BlueField's high-speed port's MAC addresses using the following Redfish command over the 1GbE OOB port to the BlueField BMC:

```

curl -k -u root:<password> -H "Content-Type: application/octet-stream" -X GET
https://<bmc_ip>/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/NetworkDevice
{
  "@odata.id":
"/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/NetworkDeviceFunctions/eth

```

```
"@odata.type": "#NetworkDeviceFunction.v1_9_0.NetworkDeviceFunction",
"Ethernet": {
  "MACAddress": "02:b1:b6:12:39:05",
  "MTUSize": 1500
},
"Id": "eth0f0",
"Links": {
  "OffloadSystem": {
    "@odata.id": "/redfish/v1/Systems/Bluefield"
  },
  "PhysicalPortAssignment": {
    "@odata.id":
"/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/Ports/eth0"
  }
},
"Name": "NetworkDeviceFunction",
"NetDevFuncCapabilities": [
  "Ethernet"
],
"NetDevFuncType": "Ethernet"
}
```

© Copyright 2024, NVIDIA. PDF Generated on 09/09/2024