



Installation for DPU Mode

Table of contents

Step 1 – BlueField SoC Boots

Step 2 – BlueField BMC Boots

Step 3 – Change Default Password

Step 4 – Upgrade BlueField BMC Firmware

Update eROT Firmware

Possible Error Codes During BMC/eROT Upgrade

Update BMC Firmware

Step 5 – Upgrade BlueField Firmware Components and BSP

Redfish Interface

BFB Live Firmware Update

Tracking Installation Progress and Status

Direct SCP

Transferring BFB File

Step 6 – Verify Software Component Versions

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

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

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

Step 10 – (Optional) Disable Secure Boot

Contents:

(i) 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, refer to section "Identifying Which Mode BlueField is Currently Operating In" on the same page.

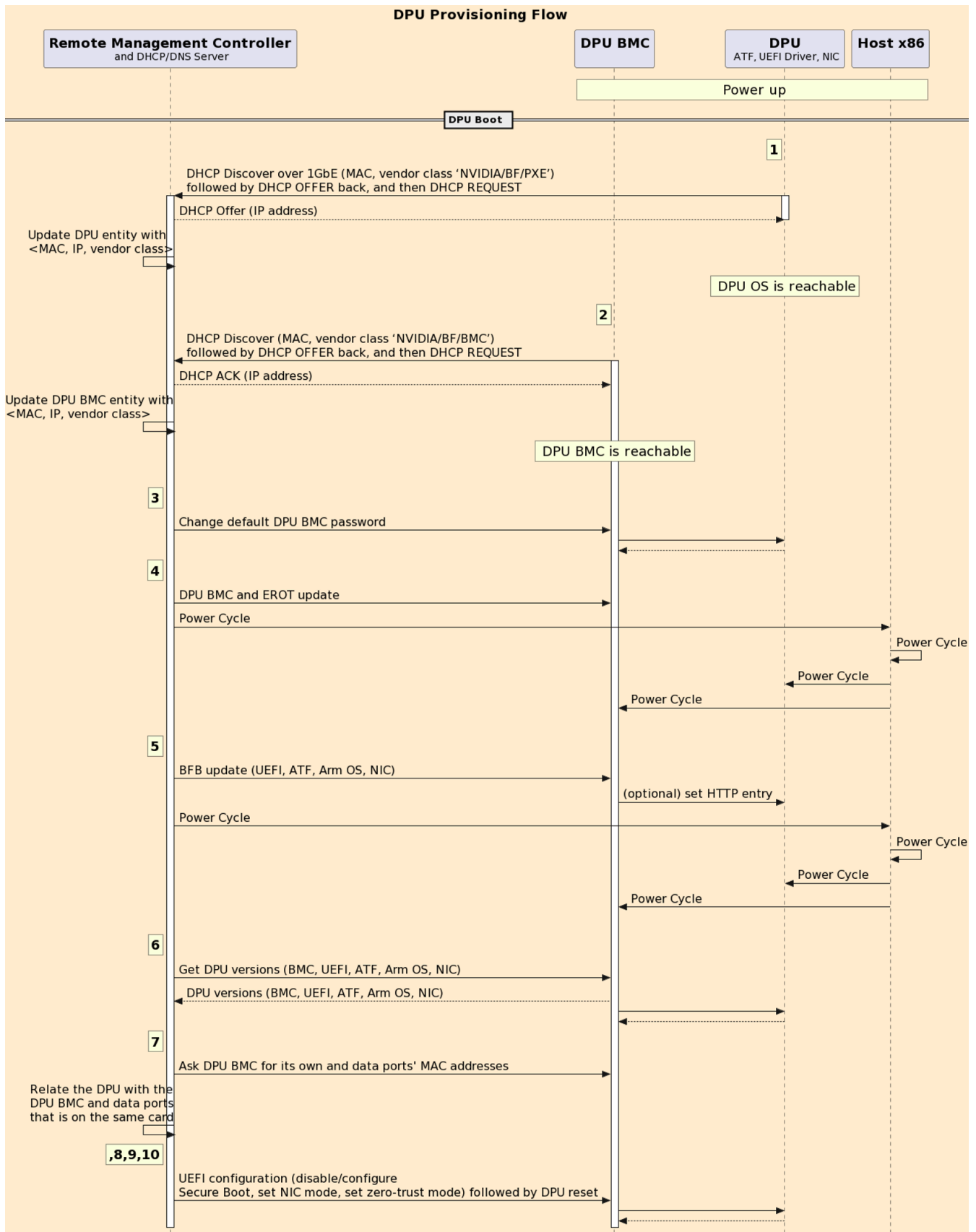
(i) Note

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.

(i) 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 > ldev-platform-13-043-bmc.bootpc: [no
cksum] BOOTP/DHCP, Reply, length 325, hops 1, xid 0xfc2acdec, secs 1,
Flags [none] (0x0000)
(example) Your-IP ldev-platform-13-043-bmc.XX.YY
(example) Server-IP l-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: "ldev-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 > ldev-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

```
|hex(char_1)-hex(char_2)|=1.
```

Examples of passwords with 5 consecutive character pairs

(invalid): `DcB a123456AbCd!` ; `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:0penBmc -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
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'
```

(i) Note

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

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

(i) 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 • 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>

Fault	Diagnosis and Possible Solution
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 • 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>

Fault	Diagnosis and Possible Solution
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.

i 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
```

Note

Upgrading the BlueField networking platform using BFB Bundle updates the NIC firmware by default. NIC firmware upgrade triggers a NIC reset flow via `m1xfwreset` in the BlueField Arm.

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

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

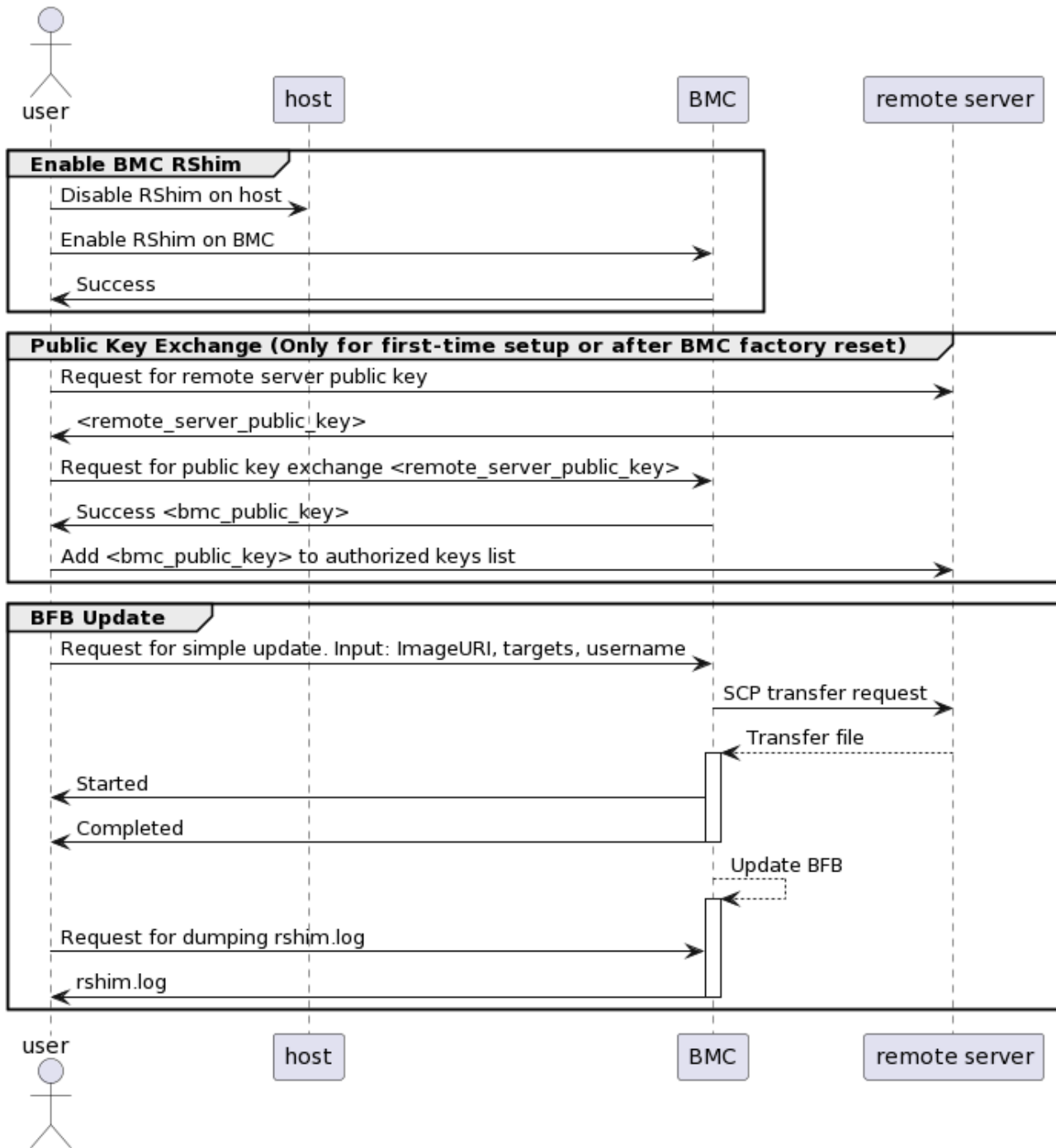
Transferring BFB File

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:

(i) 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/Nvid
```

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

```

(i) 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.

(i) 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>`).

(i) 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"

```

i 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/Action
```

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>/Monit
"TaskState": "Completed",
"TaskStatus": "OK"

```

Applying New BFB Image

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

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

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

1. Graceful shutdown the BlueField Arm OS.

Info

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

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

Info

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

4. Log into BlueField BMC via Redfish and issue a restart (BlueField BMC and CEC is applied).
5. Verify that the new BFB is running by checking its version:

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

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
```

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>`).

i 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/Cert
-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
```

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.

i 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>`).

i 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
```

BFB Live Firmware Update

To perform a BFB live firmware update:

1. Enable LFWP for the update:

```
curl -k -u root:'<password>' -X POST -d '{"LFWP": "Enabled"}'  
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Oem/Nvidia/Actio
```

2. Perform a BFB update, as specified in section "[Applying New BFB Image](#)".
3. Once the update is complete, disable LFWP:

```
curl -k -u root:'<password>' -X POST -d '{"LFWP":  
"Disabled"}'  
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Oem/Nvidia/Actio
```

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/LogServ
```

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/LogServ --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`
- `</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
```

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_Firm
    },
    {
      "@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/VL
}
}

```

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/Nvid
{
  "@odata.id":
"/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapte
  "@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/NvidiaNetworkAdapte
    }
  },
  "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
```

(i) 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.

Note

The "SecureBootEnable" property in secure boot schema takes precedence over UEFI menu BlueField Arm OS settings. This is because currently there is no separate pending setting URL and hence the value of "SecureBootEnable" property will be applied on BlueField Arm OS reboot.

- 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.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation (“NVIDIA”) makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF

ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

© Copyright 2026, NVIDIA. PDF Generated on 02/28/2026