NVIDIA BlueField BMC Software v23.10-7

Table of contents

List of Figures

About This Document

BMC software enables control and management of the baseboard management controller's (BMC) hardware components. The BMC software supports the Intelligent Platform Management Interface (IPMI).

This guide provides general information concerning the BMC on the NVIDIA® BlueField® DPUs and is intended for those who want to familiarize themselves with the functionality provided by the BMC.

⚠ **Warning**

This document is relevant for DPUs with an integrated BMC. Please refer to the Supported Platforms and Interoperability page to ascertain whether your device features an integrated BMC.

Software Download

To download product software, please refer to the BlueField software product page.

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.

Related Documentation

Document Name	Description
NVIDIA BlueField DPU Platform Operating System Documentation	This document provides product release notes as well as information on the BlueField software distribution and how to develop and/or customize applications, system software, and file system images for the BlueField platform
NVIDIA BlueField-2 Ethernet DPU User Guide	This manual describes BlueField-2 Ethernet DPU including details as to the interfaces of the board, specifications, required software and firmware, and a step-by-step plan of how to bring it up
NVIDIA BlueField-3 Ethernet DPU User Guide	This manual describes BlueField-3 Ethernet DPU including details as to the interfaces of the board, specifications, required software and firmware, and a step-by-step plan of how to bring it up
BlueField DPU Administrator Quick Start Guide	This quick start guide details the procedure for installing a brand-new NVIDIA® BlueField® DPU
NVIDIA BlueField DPU Management and Initial Provisioning	This document defines the NVIDIA-recommended method to manage NVIDIA® BlueField®-2 and BlueField®-3 DPUs, reviews BlueField DPU management interfaces, protocols, and capabilities (hardware, firmware, etc.), and explains how to use them to manage the DPU.
Redfish Data Model Specification	This document describes the architecture of IPMI design.
IPMI Architecture GitHub	This document describes the architecture of IPMI design.

Glossary

Abbreviation / Acronym	Whole Word / Description
BMC	Baseboard management controller
DPU	Data processing unit
EEPROM	Electrically Erasable Programmable Read Only Memory
FRU	Field Replaceable Unit
IPMB	Intelligent Platform Management Bus
IPMI	Intelligent Platform Management Interface
SoC	System-on-chip
SOL	Serial Over LAN
SEL	System Event Log
SDR	Sensor Data Record; Sensor Data Repository
UART	Universal Asynchronous Receiver Transmitter

Release Notes

The following pages provide information on the supported platforms, changes and new features, and reports on software known issues as well as bug fixes.

- Changes and New Features
- Supported Platforms and Interoperability
- Known Issues
- Bug Fixes in This Version
- Bug Fixes History
- Change Log History

Changes and New Features

ⓘ **Note**

For an archive of changes and features from previous releases, please refer to "Change Log History".

Changes and New Features in v23.10-7

- Bug fixes

Supported Platforms and Interoperability

Supported NVIDIA BlueField-3 DPU Platforms

SKU	**PSID**	**Description**
900-9D3B6-00CV-AA0	MT_0000000884	NVIDIA BlueField-3 B3220 P-Series FHHL DPU; 200GbE (default mode) / NDR200 IB; Dual-port QSFP112; PCIe Gen5.0 x16 with x16 PCIe extension option; 16 Arm cores; 32GB on-board DDR; integrated BMC; Crypto Enabled
900-9D3B6-00SV-AA0	MT_0000000965	NVIDIA BlueField-3 B3220 P-Series FHHL DPU; 200GbE (default mode) / NDR200 IB; Dual-port QSFP112; PCIe Gen5.0 x16 with x16 PCIe extension option; 16 Arm cores; 32GB on-board DDR; integrated BMC; Crypto Disabled
900-9D3B6-00CC-AA0	MT_0000001024	NVIDIA BlueField-3 B3210 P-Series FHHL DPU; 100GbE (default mode) / HDR100 IB; Dual-port QSFP112; PCIe Gen5.0 x16 with x16 PCIe extension option; 16 Arm cores; 32GB on-board DDR; integrated BMC;Crypto Enabled
900-9D3B6-00SC-AA0	MT_0000001025	NVIDIA BlueField-3 B3210 P-Series FHHL DPU; 100GbE (default mode) / HDR100 IB; Dual-port QSFP112; PCIe Gen5.0 x16 with x16 PCIe extension option; 16 Arm cores; 32GB on-board DDR; integrated BMC; Crypto Disabled

Self-hosted BlueField-3 DPUs

Check the following table for the SKUs of controller board :

Part Number	Description
900-9D3B6-00CV-DA0	NVIDIA BlueField-3 B3220SH E-Series FHHL Storage Controller, 200GbE (default mode) / NDR200 IB, Dual-port QSFP112, PCIe Gen5.0 x16 with x16 PCIe extension option, 16 Arm cores, 32GB on-board DDR, integrated BMC, Crypto Enabled, Tall Bracket

Part Number	Description
900-9D3C6-00CV-GA0	NVIDIA BlueField-3 B3220SH E-Series No Heatsink FHHL Storage Controller, 200GbE (default mode) / NDR200 IB, Dual-port QSFP112, PCIe Gen5.0 x16 with x16 PCIe extension option, 16 Arm cores, 48GB on-board DDR, integrated BMC, Crypto Enabled, Tall Bracket
900-9D3C6-00CV-DA0	NVIDIA BlueField-3 B3220SH E-Series FHHL Storage Controller, 200GbE (default mode) / NDR200 IB, Dual-port QSFP112, PCIe Gen5.0 x16 with x16 PCIe extension option, 16 Arm cores, 48GB on-board DDR, integrated BMC, Crypto Enabled, Tall Bracket

Supported NVIDIA BlueField-2 DPU Platforms

NVIDIA SKU	**Legacy OPN**	**PSID**	**Description**
900-9D218-0073-ST1	MBF2H512C-AESOT	MT_0000000723	BlueField-2 P-Series DPU 25GbE Dual-Port SFP56; integrated BMC; PCIe Gen4 x8; Secure Boot Enabled; Crypto Disabled; 16GB on-board DDR; 1GbE OOB management; FHHL
900-9D218-0083-ST2	MBF2H512C-AECOT	MT_0000000724	BlueField-2 P-Series DPU 25GbE Dual-Port SFP56; integrated BMC; PCIe Gen4 x8; Secure Boot Enabled; Crypto Enabled; 16GB on-board DDR; 1GbE OOB management; FHHL
900-9D208-0086-ST4	MBF2M516C-EECOT	MT_0000000728	BlueField-2 E-Series DPU 100GbE/EDR/HDR100 VPI Dual-Port QSFP56; integrated BMC; PCIe Gen4 x16; Secure Boot Enabled; Crypto Enabled; 16GB on-board DDR; 1GbE OOB management; Tall Bracket; FHHL
900-9D208-0086-SQ0	MBF2H516C-CECOT	MT_0000000729	BlueField-2 P-Series DPU 100GbE Dual-Port QSFP56; integrated BMC; PCIe Gen4 x16; Secure Boot Enabled; Crypto Enabled; 16GB on-board DDR; 1GbE OOB management; Tall Bracket; FHHL
900-9D208-0076-ST5	MBF2M516C-CESOT	MT_0000000731	BlueField-2 E-Series DPU 100GbE Dual-Port QSFP56; integrated BMC; PCIe Gen4 x16;

NVIDIA SKU	**Legacy OPN**	**PSID**	**Description**
			Secure Boot Enabled; Crypto Disabled; 16GB on-board DDR; 1GbE OOB management; Tall Bracket; FHHL
900-9D208-0076-ST6	MBF2M516C-EESOT	MT_0000000732	BlueField-2 E-Series DPU 100GbE/EDR/HDR100 VPI Dual-Port QSFP56; integrated BMC; PCIe Gen4 x16; Secure Boot Enabled; Crypto Disabled; 16GB on-board DDR; 1GbE OOB management; Tall Bracket; FHHL
900-9D208-0086-ST3	MBF2M516C-CECOT	MT_0000000733	BlueField-2 E-Series DPU 100GbE Dual-Port QSFP56; integrated BMC; PCIe Gen4 x16; Secure Boot Enabled; Crypto Enabled; 16GB on-board DDR; 1GbE OOB management; Tall Bracket; FHHL
900-9D208-0076-ST2	MBF2H516C-EESOT	MT_0000000737	BlueField-2 P-Series DPU 100GbE/EDR/HDR100 VPI Dual-Port QSFP56; integrated BMC; PCIe Gen4 x16; Secure Boot Enabled; Crypto Disabled; 16GB on-board DDR; 1GbE OOB management; Tall Bracket; FHHL
900-9D208-0076-ST1	MBF2H516C-CESOT	MT_0000000738	BlueField-2 P-Series DPU 100GbE Dual-Port QSFP56; integrated BMC; PCIe Gen4 x16; Secure Boot Enabled; Crypto Disabled; 16GB on-board DDR; 1GbE OOB management; Tall Bracket; FHHL
900-9D218-0083-ST4	MBF2H532C-AECOT	MT_0000000765	BlueField-2 P-Series DPU 25GbE Dual-Port SFP56; integrated BMC; PCIe Gen4 x8; Secure Boot Enabled; Crypto Enabled; 32GB on-board DDR; 1GbE OOB management; FHHL
900-9D218-0073-ST0	MBF2H532C-AESOT	MT_0000000766	BlueField-2 P-Series DPU 25GbE Dual-Port SFP56; integrated BMC; PCIe Gen4 x8; Secure Boot Enabled; Crypto Disabled; 32GB on-board DDR; 1GbE OOB management; FHHL
900-9D208-0076-ST3	MBF2H536C-CESOT	MT_0000000767	BlueField-2 P-Series DPU 100GbE Dual-Port QSFP56; integrated BMC; PCIe Gen4 x16; Secure Boot Enabled; Crypto Disabled; 32GB on-board DDR; 1GbE OOB management; FHHL
900-9D208-0086-ST2	MBF2H536C-CECOT	MT_0000000768	BlueField-2 P-Series DPU 100GbE Dual-Port QSFP56; integrated BMC; PCIe Gen4 x16; Secure Boot Enabled; Crypto Enabled; 32GB on-board DDR; 1GbE OOB management; FHHL

NVIDIA SKU	**Legacy OPN**	**PSID**	**Description**
900-9D218-0073-ST4	MBF2H512C-AEUOT	MT_0000000972	BlueField-2 P-Series DPU 25GbE Dual-Port SFP56; integrated BMC; PCIe Gen4 x8; Secure Boot Enabled with UEFI disabled; Crypto Disabled; 16GB on-board DDR; 1GbE OOB management
900-9D208-0076-STA	MBF2H516C-CEUOT	MT_0000000973	BlueField-2 P-Series DPU 100GbE Dual-Port QSFP56; integrated BMC; PCIe Gen4 x16; Secure Boot Enabled with UEFI disabled; Crypto Disabled; 16GB on-board DDR; 1GbE OOB management
900-9D208-0076-STB	MBF2H536C-CEUOT	MT_0000001008	BlueField®-2 P-Series DPU 100GbE Dual-Port QSFP56, integrated BMC, PCIe Gen4 x16; Secure Boot Enabled with UEFI Disabled; Crypto Disabled; 32GB on-board DDR; 1GbE OOB management; Tall Bracket; FHHL

Supported OpenBMC

- OpenBMC 2.9.0
- Linux Kernel 5.10
- U-boot 2019.04

Known Issues

ⓘ **Note**

Please make sure to also be aware of the known issues and limitations of the BSP here.

Ref #	Issue
3668925	Description: If a VLAN setup is necessary for a specific interface on the BMC, finish all other network configurations (such as DHCP/STATIC) on the interface before implementing the VLAN setting (because the VLAN inherits all configurations from the existing interface).

	Workaround:
	1. Initialize the network interface:
	`ipmitool lan set 1 ipsrc static`
	`ipmitool lan set 1 ipaddr <ip>`
	`ipmitool lan set 1 netmask <netmask>`
	`ipmitool lan set 1 defgw ipaddr <gateway-ip>`
	2. Set the VLAN:
	`ipmitool lan set 1 vlan id <vlan-id>`

	Discovered in version: 23.10
3534150	Description: The BMC and DPU utilize a shared IPMB channel for IPMI communication. If multiple requests coincide on this interface, users may encounter command failures with timeout indications.
	Workaround: Raise the retry counter for IPMItool requests by using the command "`ipmitool -R 20 *`".
	Discovered in version: 23.10
3631199	Description: If Redfish is enabled in the UEFI menu (default), then Secure Boot configuration done from Redfish overrides Secure Boot configuration done from UEFI.
	Workaround: Disable Redfish in UEFI menu and update secure boot state.
	Discovered in version: 23.10
3662417	Description: The BMC may provide incorrect bootstrap credentials to the UEFI. This would result in the failure of any BIOS configurations.
	Workaround: Perform an additional reset to the DPU.
	Discovered in version: 23.10

Ref #	Issue
3654930	Description: If the DPU BMC firmware has been upgraded from older versions (i.e., 2.8.2-x) to newer versions (i.e., 23.03 onward), it is necessary to execute a factory reset of the DPU BMC.
	Workaround: N/A
	Discovered in version: 23.03
3637527	Description: The BlueField Redfish BIOS/UEFI supports only UEFI mode for BootSourceOverrideMode. If a user configures the BootSourceOverrideMode to legacy, all override settings are disregarded by the BIOS/UEFI.
	Workaround: Set BootSourceOverrideMode to UEFI.
	Discovered in version: 23.10
3634649	Description: In the Redfish Systems/Bluefield schema, the LastResetTime attribute does not accurately capture the system reset values.
	Workaround: N/A
	Discovered in version: 23.09
3634701	Description: In the Redfish Systems/Bluefield schema, the Description attribute is of a generic type and does not specify the DPU system.
	Workaround: N/A
	Discovered in version: 23.09
3634603	Description: When the DPU operates in NIC mode, the Arm core does not load any OS. In this scenario, any BMC functionality that relies on extracting data from the OS through the IPMB channel will be unavailable or limited. including: • Firmware inventory schema • Chassis schema • Sensors
	Workaround: N/A
	Discovered in version: 23.10
3609	Description: Following a reboot of the DPU's BMC, it is necessary to wait 30 seconds to allow for the complete loading of system services before initiating a reboot of the DPU itself.

Ref #	Issue
525	Workaround: N/A
	Discovered in version: 23.09
3590634	Description: When updating the BMC's firmware, it is critical to maintain the system powered on until the update process is finished.
	Workaround: N/A
	Discovered in version: 23.09
3599824	Description: In NIC mode, the BMC's Redfish chassis schema contains only limited information about the DPU. This is because, in this mode, the OS is not available to supply the necessary information to the BMC.
	Workaround: N/A
	Discovered in version: 23.09
3605254	Description: Following a system power cycle, both the DPU and BMC boot independently which may lead to the DPU's UEFI boot process to complete before the BMC's. As a result, when attempting to establish Redfish communication, the BMC may not yet be prepared to respond.
	Workaround: Power cycle; Redfish; boot
	Discovered in version: 23.09
3388059	Description: When BlueField-2 boots and its services are loaded, there is a possibility that the IPMI over RMCP may become unresponsive due to the default timeout for commands being set to 1 second.
	Workaround: Increase the default timeout to 10 seconds when sending IPMI RMCP commands using the `-N` option. Example command: `sudo ipmitool -I lanplus -C 17 -N 10 -H <BMC-IP> -U <BMC-User> -P <BMC-Password> mc info`
	Discovered in version: N/A

Bug Fixes in This Version

ⓘ **Note**

For an archive of bug fixes from previous releases, please refer to Bug Fixes History.

Ref #	Issue
3762031	Description: The AllowableValues field format did not fit the Redfish update schema.
	Discovered in version: 23.10

Bug Fixes History

Ref #	Issue
3561677	Description: It is not possible to modify the values of the BootOrder, BootOverride, and Secure Boot attributes from the UEFI menu because they are set by default to be configured from Redfish interface.
	Fixed in version: 23.09
3566036	Description: After performing BF BMC factory reset, the /home/root/.ssh directory is deleted which causes the first attempt to confirm the host identity and initiate a BFB update procedure to fail while displaying the error message: "Host is unknown"
	Fixed in version: 23.09
3587968	Description: VLAN 4040 serves as a dedicated VLAN for facilitating Redfish communication between UEFI and DPU BMC. However, if the OOB RJ45 port is connected to an unmanaged switch or hub, the VLAN traffic from VLAN 4040 may spill over into the broader LAN network which may lead the local UEFI to unintentionally communicate with a remote BMC instead of the intended local BMC.
	Fixed in version: 23.09

Ref #	Issue
3478796	Description: Rarely, it is possible for the BMC to exceed the boot timeout set by the root of trust. In such case, the RoT initiates a second reboot of the BMC, which is expected to result in a successful boot.
	Fixed in version: 23.09
3604148	Description: In the uncommon scenario where, following a system power cycle, the DPU fails to boot successfully, the BMC would be unable to retrieve network data from the DPU's operating system. This leads to an absence of information in the Redfish chassis schema, which is responsible for describing the network adapters.
	Fixed in version: 23.09
3600004	Description: Description: In dual-port DPU, the DPU's Redfish schema, specifically the "chassis NetworkAdapters", will replicate the data from port 1 into port 2.
	Fixed in version : 23.09
3560559	Description: If the DPU OS's OOB interface is disabled, it may lead to an issue that results in the DPU BMC losing network connectivity. This problem arises when the UEFI enables the OOB port (e.g., PXE, Redfish), but the OS does not load the necessary services and OOB kernel driver. In this scenario, the physical link remains active despite the OS driver not functioning, causing the hardware queue to become filled. Consequently, flow control pause packets are sent to the onboard 3-port switch, which may eventually lead to the DPU BMC losing its network connectivity.
	Fixed in version : 23.09
N/A	Description: If the NIC BMC boots with non-default network configuration under /run/initramfs/rw/cow/etc/systemd/network/*, then the dedicated VLAN 4040 which supports the Redfish host interface with the UEFI BIOS device is not created.
	Fixed in version : 23.09
3554128	Description: dmidecode output does not match "ipmitool fru print" output.
	Fixed in version : 23.07
2930	Description: A power cycle of the system might result in BMC MAC change.

Ref #	Issue
671	Fixed in version: 2.8.2-34
3444360	Description: IPMI LAN print does not work in stateful DHCPv6.
	Fixed in version: 2.8.2
200767989	Description: SOL console receives a garbage message when it is connected.
	Fixed in version: 2.8.2
200748177	Description: PXE boot via OOB interface enters grub mode when cold rebooting the x86 host against BFB version 3.7.0.
	Fixed in version: 2.8.2

Change Log History

Changes and New Features in v23.10

- NVIDIA® BlueField®-3 Redfish enhancements:
    - Included phosphor-logging entry for dumping /dev/rshim/misc messages
    - Implemented Redfish-based firmware configuration for switching between BlueField DPU mode and NIC mode for BlueField-3
    - Added an OEM API for enabling/disabling BMC RShim, offering more control over this critical component
- Enhanced debuggability for the DPU BMC which includes the ability to store DPU console/serial logs for troubleshooting and analysis
- Deployment of a more restrictive firewall policy to enhance system security
- Added power-capping control capabilities from the DPU BMC, providing greater power management flexibility
- Added an OEM API for key-based authentication
- Incorporated the wget application into the BMC OS

- Enhanced the system with the ability to enable\disable the DPU OOB port using IPMI commands
- Removed DPU BMC SMBus master capabilities
- CEC1736 EC firmware upgrade to version 00.02.0152.0000 – t he boot completion timeout for CEC1736 has been increased from 2 minutes to 8 minutes in this version to ensure that the BMC completes its boot process within the allotted time. If the BMC fails to boot within that period, the CEC1736 initiates a reset of the BMC.

> ⚠ **Warning**
>
> This change may lead to undesired system behavior:
>
> - If a new BMC firmware update is in progress during this period, the CEC1736 reverts to the previous version of the BMC firmware
> - If the BMC fails to provide six boot complete indications, the CEC1736 interrupts the BMC boot process, necessitating a full reset cycle to recover the DPU BMC

Changes and New Features in v23.09

- The NCSIoMCTPoSMBus interface has been activated to facilitate communication between the DPU BMC and the NIC subsystem. This activation has introduced several enhanced functionalities to the NIC subsystem's firmware, including:
    - Configuring and retrieving the DPU's operational mode
    - Configuring and retrieving the status of the RShim
    - Retrieving the strap values of the NIC subsystem on the DPU
    - Obtaining information about the OS state
- Added the ability to control BIOS secure boot configuration through the Redfish interface

Changes and New Features in v23.07

- Allow programmatic changing of BIOS/UEFI parameters via the Redfish API
- Support UEFI HTTP boot using Redfish
- Allow programmatic mechanism for changing BIOS/UEFI boot order using Redfish
- Implemented the Certificate, CertificateLocations, and CertificateService schema in the NIC BMC, including certificate information
- Implemented Redfish-based firmware update using the SimpleUpdate SCP schema for DPU recovery
- DPU BMC indication of the reset/reboot state

Changes and New Features in v23.04-3

- Added support for BMCs of BlueField-3 DPUs
- Add support for Serial Console Redirection
- Added Redfish service with the underlying schemas:
    - Redfish chassis schema to represent the DPU chassis elements including:
        - /redfish/v1/Chassis/Card1
        - /redfish/v1/Chassis/Bluefield_BMC
        - /redfish/v1/Chassis/Bluefield_ERoT
    - Redfish sensor schema:
        - /redfish/v1/Chassis/Card1/Sensors/
    - NetworkAdapter schema representing a physical network adapter capable of connecting to a computer network:
        - /redfish/v1/Chassis/Card1/NetworkAdapters

- NetworkDeviceFunction schema representing a logical interface that a network adapter exposes:
  - /redfish/v1/Chassis/Card1/NetworkAdapters/{NetworkAdapter}/NetworkDeviceFunctions/
- Port schema containing properties that describe a port of a switch, controller, chassis, or any other device that could be connected to another entity:
  - /redfish/v1/Chassis/Card1/NetworkAdapters/{NetworkAdapter}/Ports
- Management subsystem schema:
  - /redfish/v1/Managers/Bluefield_BMC
- Updated service and the properties that affect the service itself for Redfish implementation:
  - /redfish/v1/UpdateService
- Redfish FirmwareInventory schema:
  - /redfish/v1/UpdateService/FirmwareInventory
- Redfish log service:
  - /redfish/v1/Managers/Bluefield_BMC/LogServices
- Redfish user account for the system manager:
  - /redfish/v1/AccountService
  - /redfish/v1/AccountService/Roles
  - /redfish/v1/SessionService/Sessions
- Redfish session service properties:
  - /redfish/v1/SessionService
- Redfish task service:
  - /redfish/v1/TaskService

Changes and New Features in 2.8.2-34

- Updated LLDPAD to be enabled by default

Changes and New Features in 2.8.2

- First software GA release

BlueField BMC Software Overview

The BMC node enables remote power cycling, board environment monitoring, NVIDIA® BlueField® SoC temperature monitoring, board power and consumption monitoring, and individual interface resets. The BMC also supports the ability to push a bootstream to the BlueField. It is recommended to manage the DPU using Redfish commands. However, IPMI commands and sysfs monitoring infrastructure are available as well .

Important

Make sure to log into the BMC first and change the global default password to prevent malicious attackers from hacking your system.

The procedures described in this manual assume that you have already installed and powered on your device according to the instructions in the DPU's specific hardware guide.

- Support for IPMI 2.0 (v1.1) Standards
  - Thermal control – access to all relevant temperature sensors, fan control
  - System management – power state control, power on/off, reboot/reset
  - Environmental monitoring – voltage/current/power
  - Serial over LAN (SOL)
  - RMCP/RMCP+
  - Event log management

    - Event alerting
    - VLAN support
- Support for DMTF Standards
    - Redfish specification (DSP0266)
    - Network Controller Sideband Interface (NC-SI) Specification (DMTF DSP0222)
- Support for BMC image update

Connecting to BMC Interfaces

BMC Management Interface

The BMC has a separate Ethernet interface which provides network connection for management traffic to the BMC. The NVIDIA® BlueField® DPU's bracket has an RJ45 port labeled "MGMT" which is the management interface port. The management port is configured with auto-negotiation capabilities by default (100MbE to 1GbE).

The BMC interface `eth0` is the management interface, so any information displayed by `ifconfig eth0` pertains to the management interface. The MAC address to be used for `eth0` is pre-programmed in the BMC FRU EEPROM and can be found on the DPU's board label. By default, the IP address used for `eth0` is acquired via DHCP but can be configured differently.

Changing Default Password

When initially logging into the system, it is mandatory to update the default BMC password, `0penBmc`. The DPU BMC offers two methods/interfaces for changing the password:

- SSH/serial:

  To change the password, connect to the BMC via SSH/serial and log in using the root user and the default password. Upon logging in, you are prompted with the following:

  ```
  dpu-bmc login: root
  Password: <Type default password>
  You are obliged to immediately change your password (mandatory for
  administrators).
  Changing the root password.
  ```

```
Current password: <Retype the default password>
New password: <Type the new password according to the above rules>
Retype the new password: <Retype the new password>
```

- Redfish:

  The Redfish user management interface may be used to configure the new password. The following Redfish command can be employed to alter the default password:

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

The new password must comply with the following policy parameters:

- Minimum length: 13
- Maximum length: 20
- Minimum number of upper-case characters: 1
- Minimum number of lower-case characters: 1
- Minimum number of digits: 1
- Minimum number of special characters: 1

> **Note**
>
> List of special characters:
>
> - `$` (dollar sign)
> - `%` (percent sign)
> - `^` (caret/circumflex)
> - `&` (ampersand)

  - * (asterisk)
  - - (minus)
  - + (plus)
  - = (equal)
  - | (pipe)
  - ~ (tilde)
  - _ (underscore)
  - , (comma)
  - . (period/full stop)
  - ; (semicolon)
  - : (colon)
  - " (quotation mark)
  - ' (apostrophe)
  - / (forward slash)
  - \ (backslash)

- Maximum number of consecutive character pairs: 4

> **Note**
>
> Two characters are consecutive if |hex(char_1)-hex(char_2)|=1.
>
> Examples of passwords with 5 consecutive character pairs (invalid): DcBa123456AbCd!; ab1XbcYcdZdeGef!; Testing_123abcgh!.

The following is a valid example password:

- HelloNvidia3D!

> ⚠ **Warning**
>
> The root account locks after four consecutive failed attempts and automatically unlocks after 10 minutes.

Account Service

The Redfish root user can inquire about and modify the applied account policies, which encompass settings such as the number of consecutive login attempts permitted and the time period for which the system will remain locked.

The following Redfish command provides the current settings:

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET
https://10.237.53.58/redfish/v1/AccountService
```

Example output:

```
{
"@odata.id": "/redfish/v1/AccountService",
"@odata.type": "#AccountService.v1_10_0.AccountService",
"AccountLockoutDuration": 600,
"AccountLockoutThreshold": 4,
"Accounts": {
"@odata.id": "/redfish/v1/AccountService/Accounts"
},
"ActiveDirectory": {
"Authentication": {
```

```
"AuthenticationType": "UsernameAndPassword",
"Password": null,
"Username": ""
},
"LDAPService": {
"SearchSettings": {
"BaseDistinguishedNames": [
""
],
"GroupsAttribute": "",
"UsernameAttribute": ""
}
},
"RemoteRoleMapping": [],
"ServiceAddresses": [
""
],
"ServiceEnabled": false
},
"Description": "Account Service",
"Id": "AccountService",
"LDAP": {
"Authentication": {
"AuthenticationType": "UsernameAndPassword",
"Password": null,
"Username": ""
},
"Certificates": {
"@odata.id": "/redfish/v1/AccountService/LDAP/Certificates"
},
"LDAPService": {
"SearchSettings": {
"BaseDistinguishedNames": [
""
],
"GroupsAttribute": "",
"UsernameAttribute": ""
}
},
"RemoteRoleMapping": [],
"ServiceAddresses": [
""
],
"ServiceEnabled": false
},
"MaxPasswordLength": 20,
"MinPasswordLength": 13,
"Name": "Account Service",
"Oem": {
"OpenBMC": {
"@odata.id": "/redfish/v1/AccountService#/Oem/OpenBMC",
"@odata.type": "#OemAccountService.v1_0_0.AccountService",
"AuthMethods": {
"BasicAuth": true,
"Cookie": true,
"SessionToken": true,
"TLS": true,
"XToken": true
}
}
},
"Roles": {
"@odata.id": "/redfish/v1/AccountService/Roles"
},
"ServiceEnabled": true
}
```

By default, if a user attempts to log into the system with an incorrect password four times in a row, their account is locked for 600 seconds. Afterwards, the user is allowed another opportunity to log in with the correct credentials. If the user fails to log in again, the account is immediately locked for an additional 600 seconds. If the user logs in successfully, the counter of consecutive login failures is reset.

The patch command may be used to modify the default policy settings. The following example illustrates how to alter the number of allowed consecutive login attempts into the system.

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X PATCH
https://<IP>/redfish/v1/AccountService -d '{"AccountLockoutThreshold" : 10}'
```

Note

For a comprehensive understanding of the schema, please refer to the DMTF definition of the AccountService.v1_10_0.AccountService schema.

If an account becomes inaccessible, users may check the system's status using the Redfish interface using the following GET operation:

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET
https://<IP>/redfish/v1/AccountService
```

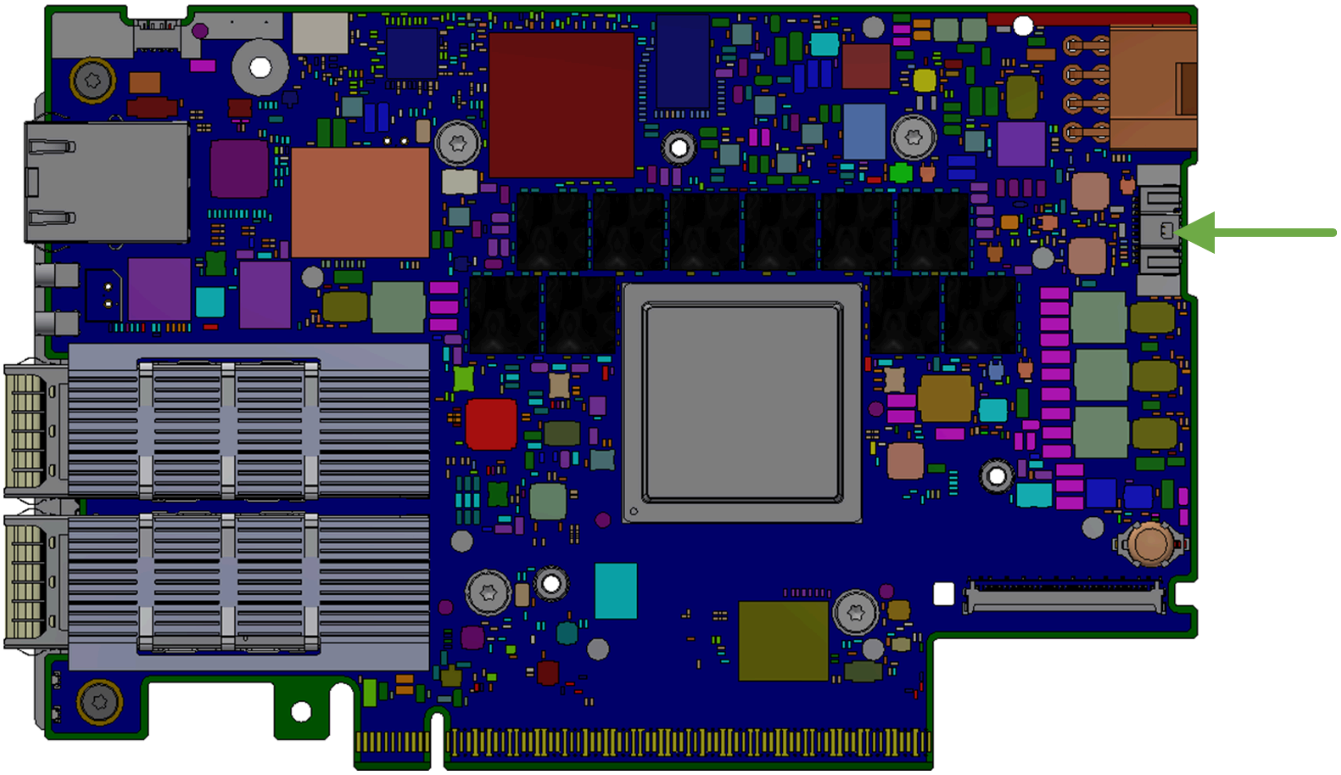
Example output:

```
{
"error": {
"@Message.ExtendedInfo": [
{
"@odata.type": "#Message.v1_1_1.Message",
"Message": "While accessing the resource at '/redfish/v1/AccountService', the service received an
authorization error 'Account temporarily locked out for 600 seconds due to multiple authentication
failures'.",
"MessageArgs": [
"/redfish/v1/AccountService",
"Account temporarily locked out for 600 seconds due to multiple authentication failures"
],
"MessageId": "Base.1.15.0.ResourceAtUriUnauthorized",
"MessageSeverity": "Critical",
"Resolution": "Ensure that the appropriate access is provided for the service in order for it to access the URI."
}
],
"code": "Base.1.15.0.ResourceAtUriUnauthorized",
"message": "While accessing the resource at '/redfish/v1/AccountService', the service received an authorization error 'Account temporarily locked out for 600 seconds due to multiple authentication failures'."
}
}
```

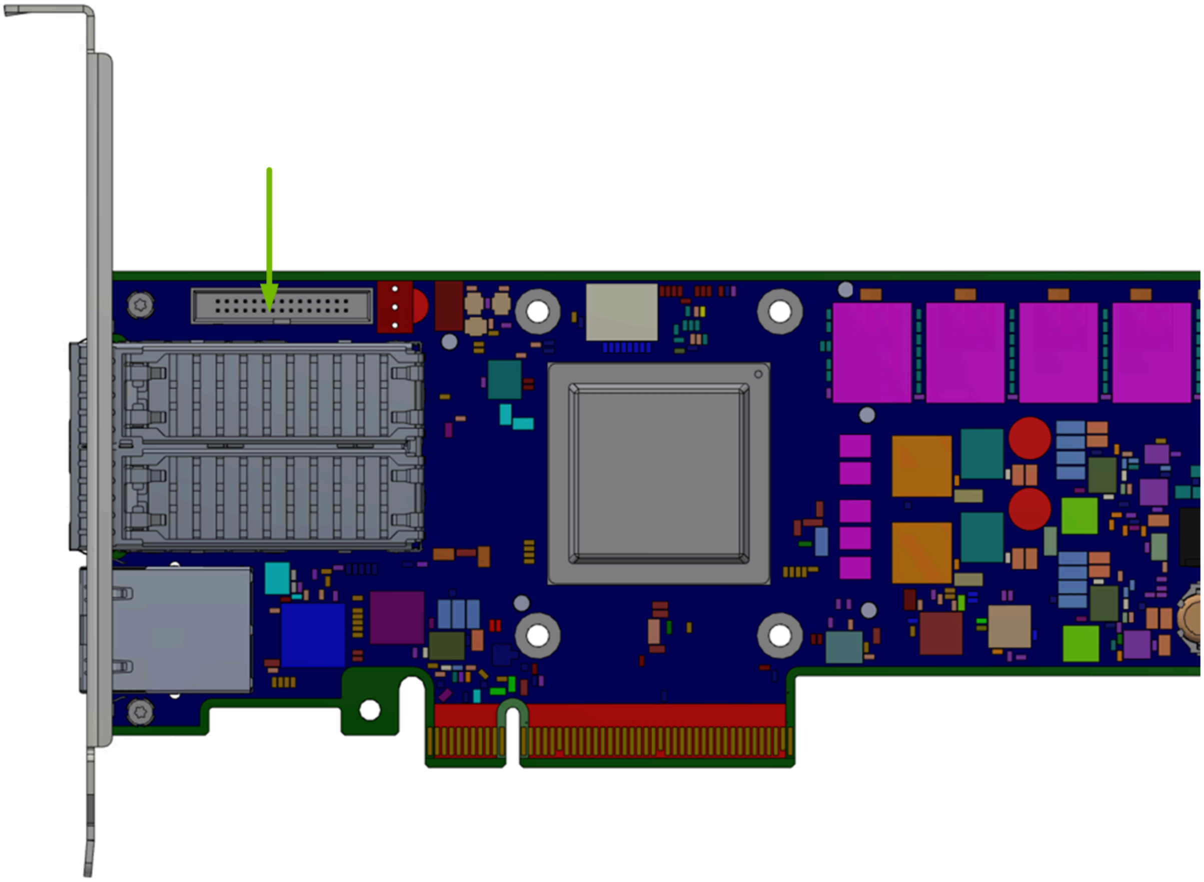
BMC Console Interface

The BMC UART1 console is available on the IO panel. The BMC is connected to a 20-pin connector for BlueField-3 or 30-pin connector for BlueField-2 which allows the Linux console to be monitored.

BlueField-3 BMC Connector



BlueField-2 BMC Connector



Network Configuration

> **Important**
>
> Do not manually modify the network configuration file `/etc/systemd/network/00-bmc-eth0.network`.

There are two ways of configuring the network interfaces:

- Dynamic (DHCP)
- Static

See section "Network Protocol Support" for more details.

BMC USB Port

This section describes the use cases for the BMC USB port. Note that only BMC Linux has access to the USB port and its feature set. There is no access to BMC USB port while running u-boot.

⚠ **Warning**

Due to a hardware bug in AST2500, the USB interface is only able to work at USB 1.0 speeds.

⚠ **Warning**

Storage device support on this port has only been validated with USB flash drives.

Providing Removable Storage via USB Stick

Once a USB stick is plugged in to the BMC's USB port, issue the command `lsusb` and/or check the dmesg log to see if the USB stick has been detected. The successful insertion of a USB stick will create a device under `/dev` called `sda` (or `sdb`), and a mountable partition `/dev/sda1`. To mount the USB stick as a filesystem, just issue the command "`mount /dev/sda1 /mnt`" to mount it at `/mnt`. The command "`umount /mnt`" unmounts the device.

System Management

This section contains the following pages:

- Platform Management Interface
- Common Configurations
- Update and Recovery
- Monitoring
- DPU Chassis
- Reset Control
- BMC and BlueField Logs
- Power Capping
- Serial Over LAN (SOL)
- Upgrading DPU Using BFB
- Vendor Field Mode
- OOB Network 3-Port Switch Control
- Serial Redirect Mode

Platform Management Interface

The NVIDIA® BlueField® DPU provides management interfaces to the BMC and the BlueField device.

Redfish Management Interface

The DPU's BMC provides a standard DMTF Redfish management interface, which is accessible via an HTTPS RESTful interface. This Redfish interface enables users to inquire about and configure the system:

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

Intelligent Platform Management Interface

The BMC, based on the IPMI standard, supports both out-of-band (OOB) dedicated interfaces, and a serial port to access the CLI of the BMC.

External Host Retrieving Data from BMC Via UART

If an external host is connected and logged into the BMC via UART, IPMI commands can be issued to fetch information from the BMC as follows:

```
ipmitool <ipmitool_arguments>
```

External Host Retrieving Data from BMC Via LAN

The BMC is connected to an external host server via LAN. IPMItool commands may be issued from the external server to retrieve information from the BMC as follows:

```
ipmitool -C 17 -I lanplus -H <bmc_ip> -U ADMIN -P ADMIN <ipmitool_arguments>
```

Common Configurations

This section contains the following pages:

- BIOS Secure Boot Configuration
- BIOS Configuration

Update and Recovery

This section contains the following pages:

- System Inventory
- Boot Configuration

Monitoring

This section contains the following pages:

- FRU Reading
- System Event Log
- Retrieving Data from BlueField Via IPMB
- BMC Sensor Data

DPU Chassis

The Redfish chassis schema provides a structured and standardized way to represent essential information about the physical infrastructure of computing systems (the DPU), offering valuable insights for system administrators, data center operators, and management software developers.

The NVIDIA® BlueField® DPU chassis encompasses all system components, which include the Bluefield_BMC, Bluefield_ERoT, and Card1 (which represents the Bluefield).

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

Output example:

```
{
"@odata.id": "/redfish/v1/Chassis",
"@odata.type": "#ChassisCollection.ChassisCollection",
"Members": [
{
"@odata.id": "/redfish/v1/Chassis/Bluefield_BMC"
},
{
"@odata.id": "/redfish/v1/Chassis/Bluefield_ERoT"
},
{
"@odata.id": "/redfish/v1/Chassis/Card1"
}
],
"Members@odata.count": 3,
"Name": "Chassis Collection"
}
```

Chassis Card1

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET
https://<bmc_ip>/redfish/v1/Chassis/Card1
```

Output example:

```
{
"@odata.id": "/redfish/v1/Chassis/Card1",
"@odata.type": "#Chassis.v1_21_0.Chassis",
"Actions": {
"#Chassis.Reset": {
"@Redfish.ActionInfo": "/redfish/v1/Chassis/Card1/ResetActionInfo",
"target": "/redfish/v1/Chassis/Card1/Actions/Chassis.Reset"
}
},
..
"ChassisType": "Card",
"EnvironmentMetrics": {
"@odata.id": "/redfish/v1/Chassis/Card1/EnvironmentMetrics"
},
"Id": "Card1",
"Links": {
"ComputerSystems": [
{
"@odata.id": "/redfish/v1/Systems/Bluefield"
}
],
"Contains": [
{
"@odata.id": "/redfish/v1/Chassis/Bluefield_ERoT"
},
{
"@odata.id": "/redfish/v1/Chassis/Bluefield_BMC"
}
],
"ManagedBy": [
{
"@odata.id": "/redfish/v1/Managers/Bluefield_BMC"
}
]
},
"Manufacturer": "Nvidia",
"Model": "Bluefield 3 SmartNIC Main Card",
"Name": "Card1",
"NetworkAdapters": {
"@odata.id": "/redfish/v1/Chassis/Card1/NetworkAdapters"
},
"PCIeDevices": {
"@odata.id": "/redfish/v1/Chassis/Card1/PCIeDevices"
```

```
},
"PCIeSlots": {
"@odata.id": "/redfish/v1/Chassis/Card1/PCIeSlots"
},
"PartNumber": "900-9D3B4-00EN-EAB ",
"Power": {
"@odata.id": "/redfish/v1/Chassis/Card1/Power"
},
"PowerState": "On",
"PowerSubsystem": {
"@odata.id": "/redfish/v1/Chassis/Card1/PowerSubsystem"
},
"SKU": "",
"Sensors": {
"@odata.id": "/redfish/v1/Chassis/Card1/Sensors"
},
"SerialNumber": "MT2245X00175 ",
"Status": {
"Conditions": [],
"Health": "OK",
"HealthRollup": "OK",
"State": "Enabled"
},
"Thermal": {
"@odata.id": "/redfish/v1/Chassis/Card1/Thermal"
},
"ThermalSubsystem": {
"@odata.id": "/redfish/v1/Chassis/Card1/ThermalSubsystem"
},
"TrustedComponents": {
"@odata.id": "/redfish/v1/Chassis/Card1/TrustedComponents"
},
"UUID": ""
}
```

Chassis Card1 NetworkAdapters

The NetworkAdapters schema specifically aims to standardize NIC management and representation. This schema includes a collection of NvidiaNetworkAdapter where each element holds the following fields:

- Ports

  The following is an example of the network port associated with `eth0`. Note that the naming conventions may differ depending on your device configuration.

  ```
  curl -k -u root:'PASSWORD' -H 'Content-Type: application/json' -X GET
  https://<IP>/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/
  ```

  Example output:

  ```
  {
  "@odata.id":
  "/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/Ports/eth0
  "@odata.type": "#Port.v1_6_0.Port",
  "CurrentSpeedGbps": 200,
  "Id": "eth0",
  "LinkNetworkTechnology": "Ethernet",
  "LinkStatus": "LinkUp",
  "Name": "Port"
  }
  ```

- NetworkDeviceFunctions

  The following is an example of the network device function for `eth0f0` (i.e., `eth0` function 0). Note that the naming conventions may differ depending on your device configuration.

  ```
  curl -k -u root:'PASSWORD' -H 'Content-Type: application/json' -X GET
  https://<IP>/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/
  ```

Example output:

```
{
"@odata.id":
"/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/NetworkDeviceFunctions/eth
"@odata.type": "#NetworkDeviceFunction.v1_9_0.NetworkDeviceFunction",
"Ethernet": {
"MACAddress": "02:8e:00:2d:4f:f8",
"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"
}
```

Reset Control

Reset Control Using Redfish

Issue the following command from the BMC to get the power status of the DPU:

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

Example output:

```
{
...
"PowerRestorePolicy": "AlwaysOn",
"PowerState": "On",
...
}
```

Hard Reset of BlueField DPU (Arm Cores and NIC Subsystem)

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

Example output:

```
{
"@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"
}
]
}
```

Hard Reset of BlueField Arm Cores

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST
https:/<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Actions/Manager.Reset -d
'{"ResetType" : "ForceRestart"}'
```

Example output:

```
{
"@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"
}
]
}
```

Reset Control Using IPMI

BMC supports reset control of NVIDIA® BlueField® through the GPIOs connected to the BMC.

Issue the following command from the BMC to get the power status of the DPU:

```
ipmitool chassis power status
```

To perform a reset of the DPU, use the following commands:

Description	Command
Hard reset of BlueField DPU (Arm cores and NIC)	`ipmitool chassis power cycle`

Description	Command
Hard reset of BlueField Arm cores	`ipmitool chassis power reset`

⚠ **Warning**

Hard reset of the BlueField DPU is allowed only when the host asserts:

- `PERST` signal on BlueField-2
- `All_STANDBY` signal on BlueField-3

OEM command `0xA1` is defined for additional non-standard reset controls of BlueField from BMC under the OEM NetFn group `0x30`.

NVIDIA OEM command to reset BlueField DPU:

Request	**Response**	Reset Option
• `0x32` – NetFun • `0xA1` – command • `0x00` – Req_data1 (reset option)	Completion code: • `0x00` – success • `<ipmi-error-code>` – failure	• `0x02` – soft reset of BlueField Arm cores ⓘ **Note** This reset command is only available when the DPU OS is up. • `0x03` – reset on-board 3-port switch

BMC and BlueField Logs

The BMC and NVIDIA® BlueField® logs can be collected using Redfish commands.

Two types of dumps are supported:

- BMC dump, which is a collection of logs from BMC
- System dump, which is a collection of logs from BlueField. To create a system dump, users must provide the BlueField credentials and IP address of the `tmfifo_net0` network interface.

BMC Dump Operations

The following subsections list BMC dump operations.

Create BMC Dump Task

Create a BMC dump task and gets the task ID.

> **Note**
>
> This is important for the next stages.

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

Where:

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

Get Dump Task State

Get dump task state. When TaskState is Completed, then the dump is ready for download.

```
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

Download BMC Dump

Download BMC dump after TaskState is Completed. Dump is saved in the path given to --output.

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

Where:

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

> ⓘ **Note**

After downloading, untar the file to view the logs.

Delete All Dump Entries

Clear all log dump entries.

```
sudo curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET
https://<ip_address>/redfish/v1/ Managers/Bluefield_BMC
/LogServices/Dump/Actions/LogService.ClearLog
```

Where:

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

Specific log dump entry deletion can be done by using ‘curl’s DELETE instead of GET in the previous command.

System Dump Operations

The following subsections list system dump operations.

Create System Dump

Create a system dump and get task ID.

```
sudo curl -k -u root:'<password>' -d '{"DiagnosticDataType": "OEM",
"OEMDiagnosticDataType": "bf_ip=<bf_ip>;bf_username=
<bf_username>;bf_password=<bf_password>"}' -X POST
https://<ip_address>/redfish/v1/Systems/Bluefield/LogServices/Dump/Actions/LogServ
```

Where:

- <ip-address> – BMC IP address
- <password> – BMC password
- <bf_ip> – BlueField IP address
- <bf_username> – BlueField username
- <bf_password> – BlueField password

Get Dump Task State

Get dump task state. The dump is ready for download when TaskState is Completed.

```
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

Download System Dump

Download the user-specified system dump.

```
sudo curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET
https://<ip_address>/redfish/v1/Systems/Bluefield/LogServices/Dump/Entries/<entry_
--output </path/to/tar/system_dump.tar.xz>
```

Where:

- <ip-address> – BMC IP address
- <password> – BMC password
- <entry_id> – The entry ID of the dump can be found in redfish/v1/Managers/Bluefield_BMC/LogServices/Dump/Entries/
- </path/to/tar/system_dump.tar.xz> – path to download the log dump system_dump.tar.xz

> **Note**
>
> After downloading, untar the file to view the logs.

Delete All Dump Entries

Clear all log dump entries.

```
sudo curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET
https://<ip_address>/redfish/v1/Systems/Bluefield/LogServices/Dump/Actions/LogServ
```

Where:

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

> **Note**

Specific log dump entry deletion can be done by using curl's DELETE instead of GET in the previous command.

The downloaded dump tar must be extracted to get the logs for BMC or BlueField.

Upon creating a dump, please allow the system ~5 mins to prepare the dump. The created dump will appear on the dump list when the system finishes dump creation. The created dump can be downloaded from the BMC using the `retrieve` command.

BlueField Console Log

BMC captures the DPU console output and stores it in the BMC dump. Refer to section "BMC Dump Operations" for getting the log files in BMC dump.

Users may also check the log in `/run/log/dpulogs/`. The log is rotated if it is larger than 1M or older than 24 hours. The oldest console output is overwritten as new data is added.

Power Capping

ⓘ **Note**

Power capping is supported on NVIDIA® BlueField®-3 only.

It is possible to adjust the system for reduced power consumption using the BMC. It is important to note that changes to power capping configuration only takes effect after DPU reboot.

ⓘ **Note**

Power capping is disabled by default.

Redfish Power Capping Requests

Get General Power Capping Information

Control information:

```
curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/Chassis/Card1/Controls/PowerLimit
```

Output example:

```
{
"@odata.id": "/redfish/v1/Chassis/Card1/Controls/PowerLimit",
"@odata.type": "#Control.v1_0_0.Control",
"AllowableMax": 300,
"AllowableMin": 200,
"ControlMode": "Manual",
"ControlType": "Power",
"Id": "PowerLimit",
"Name": "System Power Control",
"PhysicalContext": "Chassis",
"SetPoint": 50,
"SetPointType": "Single",
"SetPointUnits": "%",
"Status": {
"Health": "OK",
"State": "Enabled"
}
}
```

Power consumption information:

```
curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/Chassis/Card1/PowerSubsystem
```

Output example:

```
{
"@odata.id": "/redfish/v1/Chassis/Card1/PowerSubsystem",
"@odata.type": "#PowerSubsystem.v1_1_0.PowerSubsystem",
"Allocation ": {
"AllocatedWatts": 200
},
"Id": "PowerSubsystem",
"Name": "Power Subsystem",
"PowerSupplies": {
"@odata.id": "/redfish/v1/Chassis/Card1/PowerSubsystem/PowerSupplies"
},
"Status": {
"Health": "OK",
"State": "Enabled"
}
}
```

Enable/disable Power Capping

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X PATCH
https://<bmc_ip>/redfish/v1/Chassis/Card1/Controls/PowerLimit -d '{"SetPoint": 70,
"ControlMode":<"Manual"/"Disabled">}'
```

Set Power Allocation Percentage

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X PATCH
https://<bmc_ip>/redfish/v1/Chassis/Card1/Controls/PowerLimit -d '{"SetPoint":
```

```
<val>}'
```

Where val is the percentage of maximum capacity in Watts (AllowableMax).

Warning

If user configuration is lower than the minimum capacity power, then the UEFI sets the system power to minimum capacity.

IPMI Power Capping Commands

Get Power Capping Status

```
ipmitool raw 0x32 0xc4
```

Enable/disable Power Capping

```
ipmitool raw 0x32 0xc5 <val>
```

Where val:

- 0 – disable
- 1 – enable

Note

Changeable only from BMC prompt using admin account.

Get Power Capping Percentage

```
ipmitool raw 0x32 0xc8
```

Set Power Capping Percentage

```
ipmitool raw 0x32 0xc9 <val>
```

Where val is the value in percentage [0:100].

ⓘ **Note**

Changeable only from BMC prompt using admin account.

For example, if the maximum power capacity is 120 Watts, then set the system to work at 60 Watts (50%) using the following command:

```
ipmitool raw 0x32 0xc9 50
```

⚠ **Warning**

If user configuration is lower than the minimum capacity power, then the UEFI sets the system power to minimum capacity.

Get Maximum Power Capacity

```
ipmitool raw 0x32 0xc6
```

Note

Power is given in watts.

Get Minimum Power Capacity

```
ipmitool raw 0x32 0xca
```

Note

Power is given in watts.

Get Capacity Allocation

```
ipmitool raw 0x32 0xce
```

The amount of power allocated to the system in Watts.

This value indicates if user configuration was accepted or ignored by the UEFI.

Serial Over LAN (SOL)

If the external NVIDIA® BlueField® serial connection is not available to the switch (i.e., not connected), BMC software enables access to the BlueField through an internal serial connection redirected over an IP address.

SOL Redfish Commands

To establish the SOL connection, users may retrieve information from the redfish/v1/Systems/Bluefield schema. Inside the SerialConsole properties (SSH, IPMI), there are various methods that a client can utilize to initiate a serial session with the host through its manager.

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET
https://<bmc_ip>/redfish/v1/Systems/Bluefield
```

Example output:

```
{
...
"SerialConsole": {
"IPMI": {
"ServiceEnabled": true
},
"MaxConcurrentSessions": 15,
"SSH": {
"HotKeySequenceDisplay": "Press ~. to exit console",
"Port": 2200,
"ServiceEnabled": true
}
},
...
}
```

Based on the information provided, it is possible to establish a connection to the system's serial interface using the configured settings. In the following example, an SSH connection is utilized to connect to the system's serial interface:

```
ssh <bmc_ip> -p <port-number>
```

The port number can be obtain from the `SerialConsole` schema. In this example, that would be port 2200.

SOL IPMI Commands

To connect to serial-over-LAN use the following IPMI command from an external server:

```
ipmitool -C 17 -I lanplus -H <ip-address-of-bmc > -U ADMIN -P ADMIN sol activate
```

For example:

```
ipmitool -C 17 -I lanplus -H 10.10.10.10 -U ADMIN -P ADMIN sol activate
[SOL Session operational. Use ~? for help]

Poky (Yocto Project Reference Distro)

2.3.1 bluefield /dev/ttyAMA0

bluefield login:
```

The IPMI SOL commands are listed in the following table:

No.	Function	Command	Description

1	Get SOL info	`ipmitool sol info`
		`ipmitool sol info 1`
			Get SOL configuration data

2	Enable SOL access	`ipmitool sol set set-in-progress set-complete 1`
		`ipmitool sol set enabled true 1`
			Enable the properties to be set via set-in-progress then enable SOL access

3	Activate SOL	`ipmitool -C 17 -I lanplus -U <username> -P <password> -H <ip_address> sol activate` Where: • -U – BMC username • -H – BMC IP address • -P – BMC password	Activate SOL access to the BlueField console
4	Deactivate SOL	`ipmitool -C 17 -I lanplus -U <username> -P <password> -H <ip_address> sol deactivate`	Deactivate SOL access to the BlueField console

⚠ **Warning**

SOL feature can be used even if BlueField is configured to use UART1/ttyAMA1.

Upgrading DPU Using BFB

Network Connection from BMC to BlueField DPU

By default, the BMC and BlueField interfaces are configured as follows (static IPs and MACs):

	BMC	BlueField
Interface Name	"tmfifo_net0"	"tmfifo_net0"
MAC Address	00:1A:CA:FF:FF:02	00:1A:CA:FF:FF:01
IP Address	192.168.100.1	192.168.100.2

Enable RShim on DPU BMC

1. Disable RShim on the host. Run the following on the host:

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

> **Note**
>
> If the RShim driver is not installed on the host, this step can be skipped.

2. Enable RShim on the BMC using the Redfish interface:

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X PATCH -d '{
"BmcRShim": {
"BmcRShimEnabled": true
}
}' https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Oem/Nvidia
```

3. Check the current `BmcRShimEnabled` value and wait until it changes to `true`:

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X GET
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Oem/Nvidia
```

> **Note**
>
> This may take up to 8 seconds. If the `BmcRShimEnabled` value does not change, disable BMC RShim by setting the value to `false` then repeating steps 1-3.

Deploying BlueField Software Using BFB from BMC

To update the software on the BlueField DPU, the DPU must be booted up without mounting the eMMC flash device. This requires an external boot flow where a BFB (which includes ATF, UEFI, Arm OS, NIC firmware, and initramfs) is pushed from an external host via USB or PCIe. On BlueField DPUs with an integrated BMC, the USB interface is internally connected to the BMC and is enabled by default. Therefore, you must verify that the RShim driver is running on the BMC. This provides the ability to push a bootstream over the USB interface to perform an external boot.

Changing Default Credentials Using bf.cfg

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

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

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

1. Create password hash. Run:

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

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

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

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

⚠ **Warning**

Password policy:

- Minimum password length – 8
- At least one upper-case letter
- At least one lower-case letter
- At least one numerical character

Installing BFB

The BFB installation procedure consists of the following main stages:

1. Enabling RShim on the BMC. See section "Enable RShim on DPU BMC" for instructions.

2. Initiating the BFB update procedure by transferring the BFB image using one of the following options:

    - Direct SCP

        1. Running an SCP command.

    - Redfish interface

        1. Confirming the identity of the host and BMC—required only during first-time setup or after BMC factory reset.

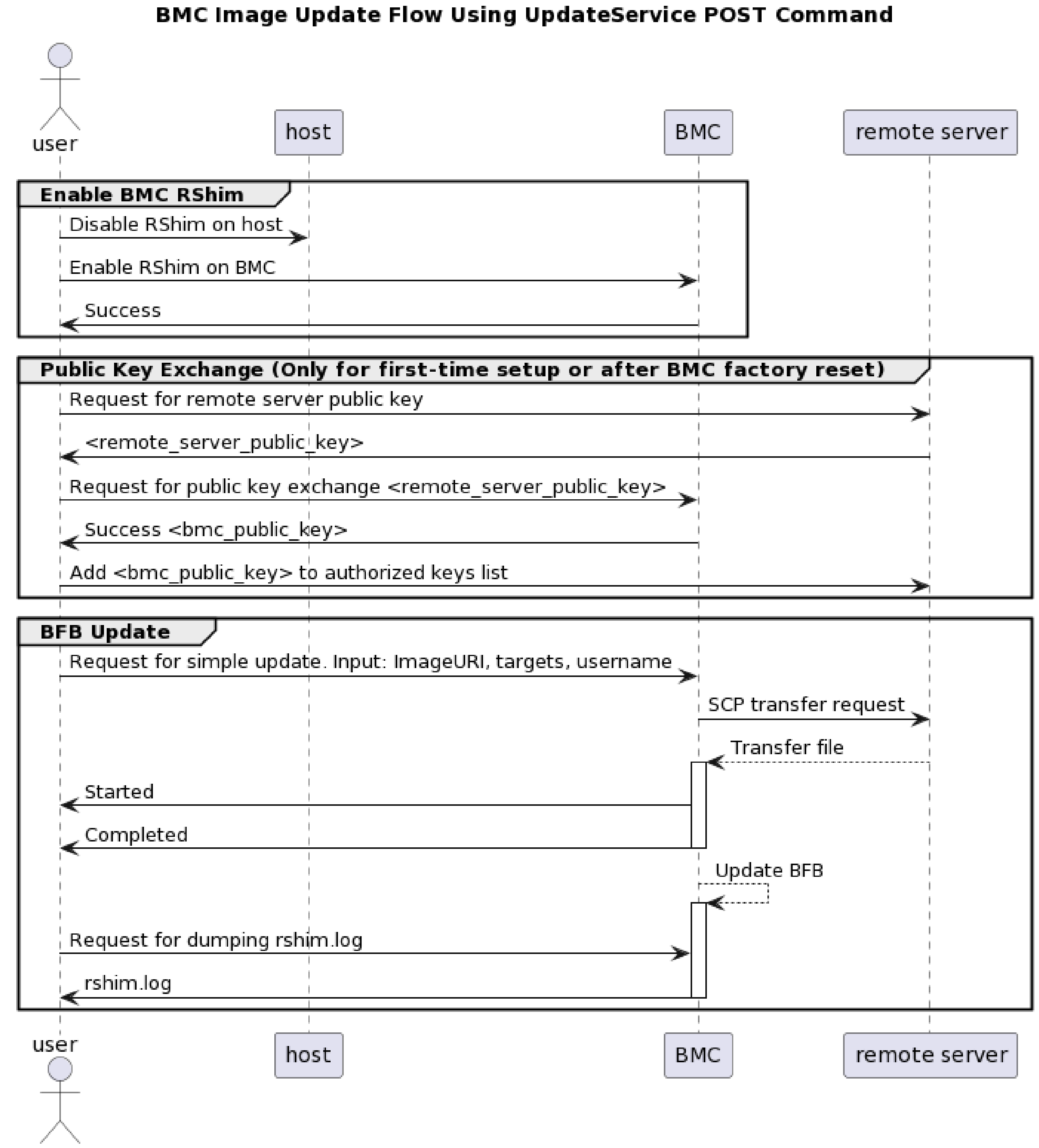
        2. Sending a Simple-Update request.

Transferring BFB Image

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

The following is a simple sequence diagram illustrating the flow of the BFB installation process.


BMC Image Update Flow Using UpdateService POST Command
user
host
BMC
remote server
Enable BMC RShim
Disable RShim on host
Enable RShim on BMC
Success
Public Key Exchange (Only for first-time setup or after BMC factory reset)
Request for remote server public key
<remote_server_public_key>
Request for public key exchange <remote_server_public_key>
Success <bmc_public_key>
Add <bmc_public_key> to authorized keys list
BFB Update
Request for simple update. Input: ImageURI, targets, username
SCP transfer request
Transfer file
Started
Completed
Update BFB
Request for dumping rshim.log
rshim.log
user
host
BMC
remote server

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

1. Confirm the identity of the remote server (i.e., host holding the BFB image) and BMC.

> **Note**
>
> Required only during first-time setup or after BMC factory reset.

1. Run the following on the remote server:

   ```
   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 public key of the host holding the BFB image from the response and provide the remote server's credentials to the DPU using the following command:

   ```
   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/NvidiaUpdateSer
   ```

   Where:

   - `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 a 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 host holding the BFB image. This enables passwordless 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"
}
]
}
```

4. If the remote server public key must be revoked, use the following command before repeating the previous step:

```
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/NvidiaUpdateServ
```

Where:

- remote_server_ip – remote server's IP address

        - bmc_ip – BMC IP address

2. Start BFB image transfer using the following command on the remote server:

```
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.SimpleUpdat
```

Note

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

Warning

This command uses SCP for the image transfer, initiates a soft reset on the BlueField and then pushes the boot stream. For Ubuntu BFBs, the eMMC is flashed automatically once the bootstream is pushed. On success, a "running" message is received with the current task ID.

Where:

    - image_uri – the image URI format should be <remote_server_ip>/<path_to_bfb>
    - username – username on the remote server
    - bmc_ip – BMC IP address

Examples:

- 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."
}
]
}
```

- 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. Wait 2 seconds and run the following on the host to track image transfer progress:

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

⚠ Warning

The transfer takes ~8 minutes for BlueField-3, and ~40 minutes for BlueField-2. 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:

1. bmc_ip – BMC IP address
2. task_id – task ID

   Troubleshooting:

   - 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"
```

ⓘ **Note**

In this case, revoke the remote server key (step 1.d.), and repeat steps 1.a. to 1.c.

- 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"
```

ⓘ **Note**

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"
```

- 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 completion of transfer of the BFB image to the DPU, the following is received:

```
{
"@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"
```

4. When the BFB transfer is complete, dump the current RShim miscellaneous messages to check the update status.

Note

> Refer to section "BMC Dump Operations" under "BMC and BlueField Logs" for information on dumping the rshim.log which contains the current RShim miscellaneous messages.

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/DPU_OS
```

Direct SCP

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

Vendor Field Mode

Vendor field mode (VFM) allows the BMC to work in a restricted mode with limited permissions.

Enabling VFM automatically performs the following on BMC:

1. Creates a new non-superuser user with username fieldmode and enables auto-login (only on the serial port) for this user.
2. Stops network services on the BMC and disables the OOB management port. This blocks all network-related operations (e.g., ssh, https, lanplus) to BMC over the Ethernet interface.
3. Disables login for the root user.

The fieldmode user can perform the following operations over UART:

- Start/stop UART tunneling to the NVIDIA® BlueField® Arm OS (i.e., OS running on the Arm core)
- Secure firmware update and track update status of BMC and CEC components
- Reboot BMC

From the BlueField Arm OS, the user `fieldmode` will be able to enable or disable VFM.

Disabling VFM automatically performs the following on BMC:

1. Enables login for the `root` user.
2. Enables network services on the BMC and the OOB management port. This re-enables all network-related operations to BMC over the Ethernet interface.

Updating BMC Firmware with Vendor Field Mode

1. Get the status of the tunnel through UART. Run the following command on the host where the BMC is connected on the UART port:

   ```
   echo -e "\\g\\@" > /dev/ttyUSBX
   ```

   Expect the following sequence of chars when the tunnel is up and running: 169 150 230.
   Expect the following sequence of chars when the tunnel is not running: 165 200.

2. If tunnel is up and running, stop the tunneling on BMC over UART.

   ```
   echo -e "\r~." > /dev/ttyUSBX
   ```

3. Transfer the BMC firmware image over UART using the XModem tool. Run the following command on the host where the BMC is connected on the UART port:

   ```
   echo -e -n "\ncd /tmp/images\n \nrz\n" > /dev/ttyUSBX
   sz -8b OTA.tar < /dev/ttyUSBX > /dev/ttyUSBX
   ```

4. Start the firmware update. Run the following command on the host where the BMC is connected on the UART port:

   ```
   echo "touch /tmp/fw-update/fwactivate" > /dev/ttyUSBX
   ```

5. To check the progress of the firmware update on the BMC, run:

   ```
   echo "cat /tmp/fw-update/fwstatus " > /dev/ttyUSBX
   ```

   Refer to section "Supported Vendor Field Mode Commands" for different firmware update values. It takes ~40 minutes to complete the BMC firmware update.

6. After a successful firmware update to activate the new firmware, reboot the BMC using the following command on the host where the BMC is connected on the UART port:

   ```
   echo "touch /tmp/fw-update/reboot" > /dev/ttyUSBX
   ```

7. Keep polling the status of the tunnel through UART to check that the BMC has booted up.

8. Check the new BMC firmware version.

   ```
   echo "cat /etc/os-release " > /dev/ttyUSBX
   ```

Updating CEC Firmware with Vendor Field Mode

⚠ **Warning**

Relevant only for BlueField-2.

1. Get the status of the tunnel through UART. Run the following command on the host where the BMC is connected on the UART port:

```
echo -e "\\g\\@" > /dev/ttyUSBX
```

Expect the following sequence of characters when the tunnel is up and running: 169 150 230.
Expect the following sequence of characters when the tunnel is not running: 165 200.

2. If the tunnel is up and running, stop the tunneling on BMC over UART:

```
echo -e "\r~." > /dev/ttyUSBX
```

3. Transfer the BMC firmware image over UART using the XModem tool. Run the following command on the host where the BMC is connected on the UART port.

```
echo -e -n "\ncd /tmp/cec_images\n \nrz\n" > /dev/ttyUSBX
sz -8b CEC.bin < /dev/ttyUSBX > /dev/ttyUSBX
```

4. To check the progress of the firmware update on the BMC, run:

```
echo "cat /tmp/cec_images progress.txt " > /dev/ttyUSBX
```

Refer to section "Supported Vendor Field Mode Commands" for different firmware update values.

5. After a successful CEC firmware update, power cycle the board or run the following on the host to activate the new firmware:

```
host# ipmitool chassis power cycle
Chassis Power Control: Cycle
```

6. Keep polling the status of the tunnel through UART to check that BMC and CEC are booted up.

Updating BMC and Glacier Firmware with Vendor Field Mode

⚠ **Warning**

Relevant only for BlueField-3.

1. Get the status of the tunnel through UART. Run the following command on the host where the BMC is connected on the UART port:

   ```
   echo -e "\\g\\@" > /dev/ttyUSBX
   ```

   Expect the following sequence of characters when the tunnel is up and running: 169 150 230.
   Expect the following sequence of characters when the tunnel is not running: 165 200.

2. If the tunnel is up and running, stop the tunneling on BMC over UART.

   ```
   echo -e "\r~." > /dev/ttyUSBX
   ```

3. Transfer the BMC or Glacier firmware image over UART using the XModem tool. Run the following command on the host where the BMC is connected on the UART port:

   ```
   echo -e -n "\ncd /tmp/images\n \nrz\n" > /dev/ttyUSBX
   sz -8b IMAGE.fwpkg < /dev/ttyUSBX > /dev/ttyUSBX
   ```

4. Start the firmware update. Run the following command on the host where the BMC is connected on the UART port:

   ```
   echo "touch /tmp/fw-update/fwactivate" > /dev/ttyUSBX
   ```

5. To check the progress of the firmware update on the BMC, run:

   ```
   echo "cat /tmp/fw-update/fwstatus " > /dev/ttyUSBX
   ```

Refer to section "Supported Vendor Field Mode Commands" for different firmware update values. It takes ~40 minutes to complete the BMC firmware update.

6. After a successful firmware update to activate the new firmware, reboot the BMC using the following command on the host where the BMC is connected on the UART port:

```
echo "touch /tmp/fw-update/reboot" > /dev/ttyUSBX
```

7. Keep polling the status of the tunnel through UART to check that the BMC has booted up.

8. Check the new BMC firmware version.

```
echo "cat /etc/os-release " > /dev/ttyUSBX
```

Supported Vendor Field Mode Commands

Operation Description	Command
Enable VFM	Run from Arm/BlueField OS and reboot NIC-BMC: `ipmitool raw 0x32 0x67 0x01`
Disable VFM	Run from Arm/BlueField OS and reboot NIC-BMC: `ipmitool raw 0x32 0x67 0x00`
Fetch VFM	Run from Arm OS: `ipmitool raw 0x32 0x68`
Get the status of the tunnel through UART	Run the following command on the host where the BMC is connected: `echo -e "\\g\\@" > /dev/ttyUSBX` Where `/dev/ttyUSBX` is the UART port number to which BMC is connected.

Operation Description	Command
	Expect the following sequence of chars when the tunnel is up and running: 169 150 230. Expect the following sequence of chars when the tunnel is not running: 165 200.
Start tunneling on BMC through UART	Run the following command on the host where the BMC is connected: `echo "touch /tmp/fw-update/uart-tunneling" > /dev/ttyUSBX` Where `/dev/ttyUSBX` is the UART port number to which BMC is connected.
Stop tunneling on BMC through UART	Run the following command on the host where the BMC is connected: `echo -e "\r~." > /dev/ttyUSBX` Where `/dev/ttyUSBX` is the UART port number to which BMC is connected.
Reboot BMC through UART	Run the following command on the host where the BMC is connected: `echo "touch /tmp/fw-update/reboot" > /dev/ttyUSBX` Where `/dev/ttyUSBX` is the UART port number to which BMC is connected.
To start/activate the BMC firmware update on BMC through UART	Run the following command on the host where the BMC is connected: `echo "touch /tmp/fw-update/fwactivate" > /dev/ttyUSBX` Where `/dev/ttyUSBX` is the UART port number to which BMC is connected.
To check the BMC firmware update status on BMC	Run the following command on the BMC: `cat /tmp/fw-update/fwstatus` Output and their values:

Operation Description	Command
	• Activating – indicates firmware update is in progress • Active – indicates firmware update succeeded • Failed – indicates firmware update failed

To check the CEC firmware update status on BMC
⚠ **Warning**
Relevant only for BlueField-2.
	Run the following command on the BMC:
	`cat /tmp/cec_images progress.txt`
	Sample output of the `progress.txt`:
	• CEC update in progress:
	`TaskState="Running"`
	`TaskStatus="OK"`
	`TaskProgress="50"`
	• CEC update completed:
	`TaskState=Firmware update succeeded.`
	`TaskStatus=OK`
	`TaskProgress=100`

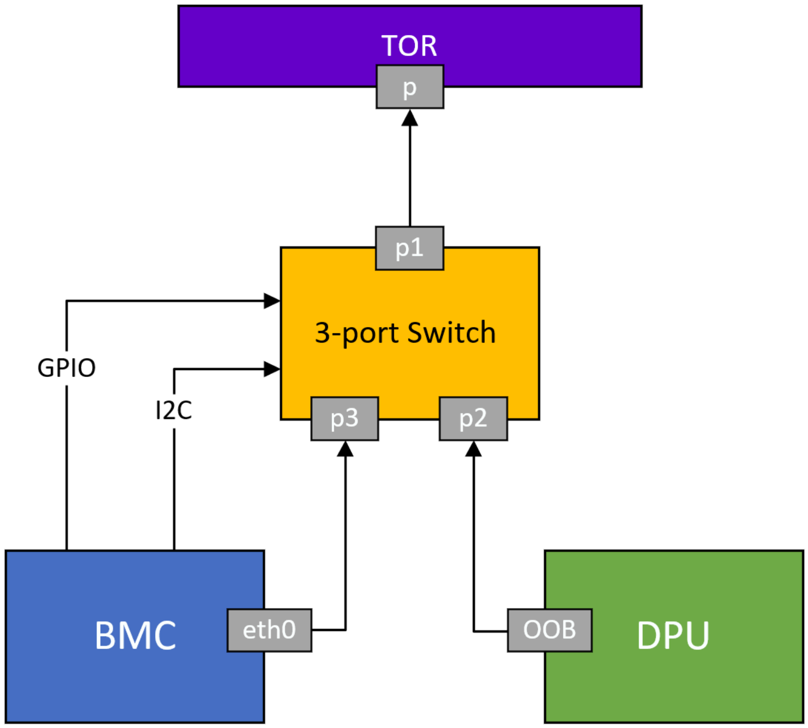
Transfer BMC firmware image for firmware update through UART	Run the following command on the host where the BMC is connected:
	`echo -e -n "\ncd /tmp/images\n \nrz\n" > /dev/ttyUSBX`
	Run the following command on the host where the BMC is connected:
	`sz -8b OTA.tar < /dev/ttyUSBX > /dev/ttyUSBX`
	Where `/dev/ttyUSBX` is the UART port number to which BMC is connected.

Operation Description	Command

Transfer CEC firmware image for firmware update through UART
⚠ **Warning**
Relevant only for BlueField-2.
	Run the following command on the host where the BMC is connected:
	`echo -e -n "\ncd /tmp/cec_images\n \nrz\n" > /dev/ttyUSBX`
	Run the following command on the host where the BMC is connected:
	`sz -8b OTA.bin < /dev/ttyUSBX > /dev/ttyUSBX`
	Where `/dev/ttyUSBX` is the UART port number to which BMC is connected.

OOB Network 3-Port Switch Control

To enable both the BMC and the Arm on the DPU to access the out-of-band (OOB) network management interface, an L2, 3-port switch has been incorporated into the system. This switch acts as a bridge, connecting the RG45 port (OOB), the BMC, and the Arm in the DPU. It is important to note that the switch is exclusively managed by the DPU's BMC through a dedicated I2C line and a GPIO signal that controls the switch's reset function.



3-Port Switch IPMI Commands

netfunc	cmd	data	Description
0x32	0x97	N/A	Get 3-port switch ports mode. On success, it returns: • 0x00 – all ports are allowed access to RJ45 • 0x01 – only BMC is allowed access to RJ45
0x32	0x98	• 0x00 – all ports are allowed access to RJ45 • 0x01 – only BMC is allowed access to RJ45	Set 3-port switch ports mode. Note: • Setting this command is only possible while the user is logged on to the BMC, this command is not supported over the network interfaces (IPMI nor Redfish) • Setting is persistent across power cycle and switch reset command

netfunc	cmd	data	Description
0x32	0xA1	0x3	Reset on-board 3-port switch

Note

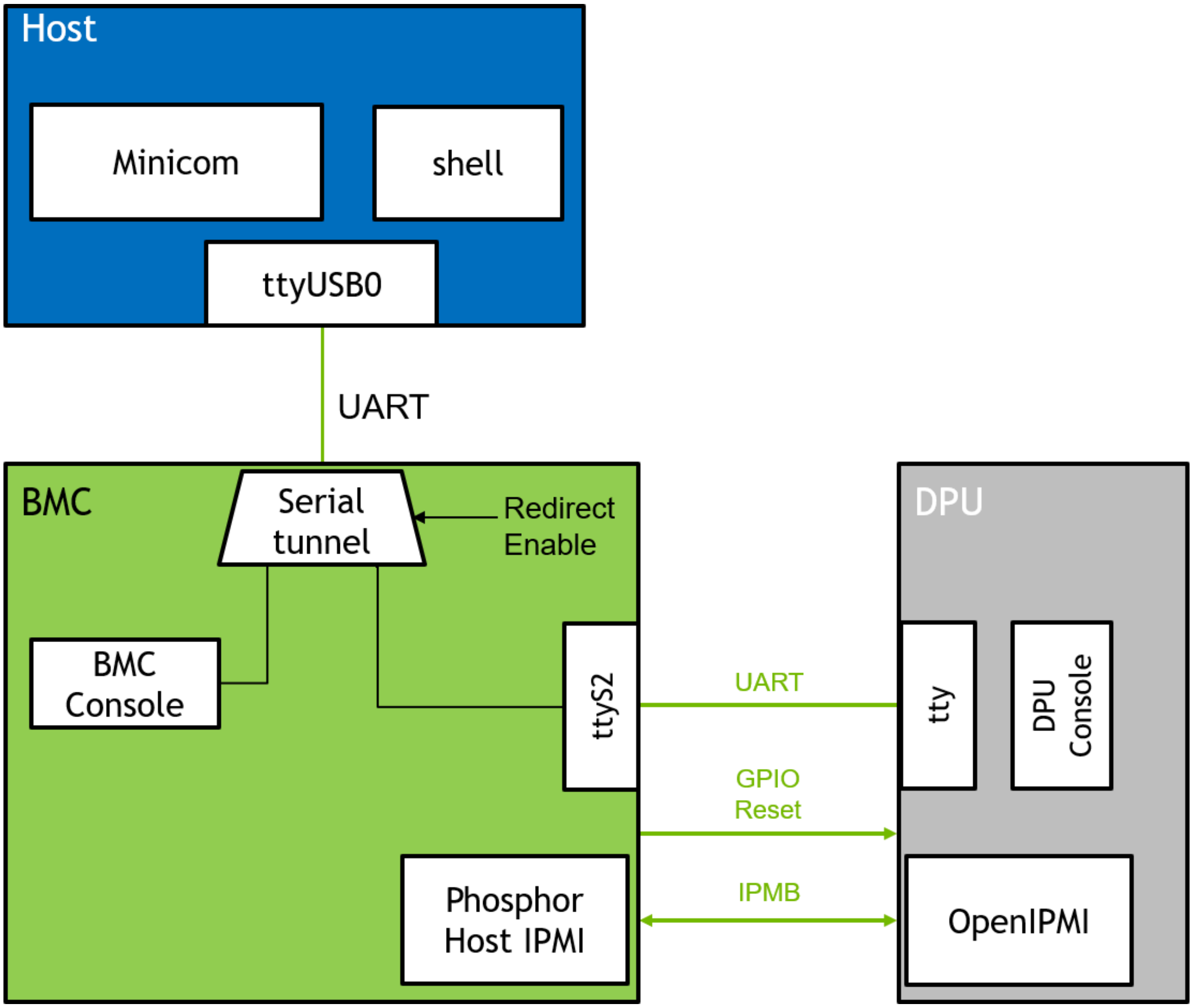
In all these use cases, the internal pathway connecting the DPU and the BMC remains operational. This enables communication between the BMC and the DPU over the internal network.

Example for disabling the OOB network of the DPU Arm:

```
#bmc> ipmitool raw 0x32 0x98 0x1
```

Serial Redirect Mode

Serial redirect mode enables the BMC to tunnel the Arm console to the external BMC console.



To enable/disable serial redirect mode:

1. Run the enable/disable serial redirect mode command from the NVIDIA® BlueField® Arm or BMC OS.
2. Run the fetch serial redirect mode command to verify the serial redirect mode's status.
3. Reboot BMC.

Enabling serial redirect mode automatically sets the following on the BMC:

1. Disables vendor field mode if enabled.
2. Enables auto login (only on the serial port) for the root user. Root user can also log in using SSH through the OOB port.

3. Enables tunneling on BMC through UART by default.

4. DPU BMC validates that BlueField is in controller mode (refer to the self-hosted SKUs), and if so, it resets (SOC_HARD_RESET) the DPU.

Disabling serial redirect mode automatically sets the following on the BMC:

1. Disables auto login (only on serial port) for the root user.

2. Disables tunneling on BMC through UART by default.

The following table lists the supported commands:

Operation	Command
Enable serial redirect mode settings to be run from the Arm or BMC OS	`ipmitool raw 0x32 0x6D 0x01`
Disable serial redirect mode settings from being run on the Arm or BMC OS	`ipmitool raw 0x32 0x6D 0x00`
Fetch serial redirect mode settings	`ipmitool raw 0x32 0x6E`
Start tunneling on BMC through UART	Run the following command on the host where BMC is connected**:** `/usr/bin/nvidia-field-mode-modifier starttunnel`
Stop tunneling on BMC through UART	Run the following command on the host where BMC is connected**:** `echo -e "\r~." > /dev/ttyUSBX` Where `/dev/ttyUSBX` is the UART port number to which the BMC is connected.

BMC Management

NVIDIA BMC is based on the OpenBMC open-software framework which builds a complete Linux image for a board management controller (BMC). It uses the Yocto project as the underlying building and distro generation framework.

The primary software components of BMC are the following:

- U-boot bootloader
- Linux kernel
- OpenBMC distro

Software Versioning

There is a software version for each of the BMC software components. You may retrieve this information by running the following for each component:

- Linux version – `uname -a` command from the Linux prompt
- OpenBMC version – `cat /etc/os-release` from the Linux prompt

Retrieving BMC Version Using Redfish

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET
https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory/BMC_Firmware
{
"@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/BMC_Firmware",
"@odata.type": "#SoftwareInventory.v1_4_0.SoftwareInventory",
"Description": "BMC image",
"Id": "BMC_Firmware",
"Name": "Software Inventory",
"RelatedItem": [],
"RelatedItem@odata.count": 0,
"SoftwareId": "",
"Status": {
"Conditions": [],
"Health": "OK",
"HealthRollup": "OK",
"State": "Enabled"
},
"Updateable": true,
"Version": "BF-23.09-1",
"WriteProtected": false
}
```

Retrieving BMC Version Using IPMI

```
# ipmitool mc info
Device ID : 1
Device Revision : 1
Firmware Revision : 23.09
IPMI Version : 2.0
Manufacturer ID : 33049
Manufacturer Name : NVIDIA
Product ID : 4 (0x0004)
Product Name : Bluefield3 BMC
Device Available : yes
Provides Device SDRs : yes
Additional Device Support :
Sensor Device
SDR Repository Device
SEL Device
FRU Inventory Device
IPMB Event Receiver
Chassis Device
Aux Firmware Rev Info :
0x10
```

```
0x01
0x00
0x00
```

Where the BMC version is composed of: [Firmware Revision]-[Aux Firmware Rev Info 2nd byte] in this example 23.9-1.

Boot Sequence Overview

1. BMC starts booting through u-boot bootloader once the power supply is powered on.
2. By default, the BMC automatically boots into Linux. To stop at the u-boot prompt, users must type the password `0penBmc` (note the use of the digit zero in `0pen`) within 5 seconds. To boot Linux from the u-boot prompt, type `boot`.
3. The BMC provides indications of its status during its operation:

Scenario	Message
At the beginning of the boot process of the u-boot	`Nvidia Bluefield BMC U-BOOT starting`
At the beginning of the OS boot process	`Nvidia Bluefield BMC Starting kernel ...`
At the login prompt	`Nvidia Bluefield BMC OS is up and running`
Upon reboot or shutdown	`Nvidia Bluefield BMC is shutting down`

4. The default password for the root user, to be typed in once Linux is booted, is `0penBmc`.

ⓘ **Note**

For information on password policy, refer to section "BMC Management Interface".

User Management

User Management Redfish Commands

General Information

General information about the BMC account services

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET
https://<IP>/redfish/v1/AccountService
```

Example output:

```
{
"@odata.id": "/redfish/v1/AccountService",
"@odata.type": "#AccountService.v1_10_0.AccountService",
"AccountLockoutDuration": 600,
"AccountLockoutThreshold": 4,
"Accounts": {
"@odata.id": "/redfish/v1/AccountService/Accounts"
},
..
"MaxPasswordLength": 20,
"MinPasswordLength": 13,
"Name": "Account Service",
"Oem": {
..
"Roles": {
```

```
"@odata.id": "/redfish/v1/AccountService/Roles"
},
"ServiceEnabled": true
}
```

List Supported User Roles

List supported user roles in the system:

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET
https://<IP>/redfish/v1/AccountService/Roles
```

Example output:

```
{
"@odata.id": "/redfish/v1/AccountService/Roles",
"@odata.type": "#RoleCollection.RoleCollection",
"Description": "BMC User Roles",
"Members": [
{
"@odata.id": "/redfish/v1/AccountService/Roles/Administrator"
},
{
"@odata.id": "/redfish/v1/AccountService/Roles/Operator"
},
{
"@odata.id": "/redfish/v1/AccountService/Roles/ReadOnly"
},
{
"@odata.id": "/redfish/v1/AccountService/Roles/NoAccess"
}
],
"Members@odata.count": 4,
"Name": "Roles Collection"
}
```

List User Accounts

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET
https://<IP>/redfish/v1/AccountService/Accounts
```

Example output:

```
{
"@odata.id": "/redfish/v1/AccountService/Accounts",
"@odata.type": "#ManagerAccountCollection.ManagerAccountCollection",
"Description": "BMC User Accounts",
"Members": [
{
"@odata.id": "/redfish/v1/AccountService/Accounts/NvdBluefieldUefi"
},
{
"@odata.id": "/redfish/v1/AccountService/Accounts/root"
}
],
"Members@odata.count": 2,
"Name": "Accounts Collection"
}
```

Create New User

Create a new user on the BMC:

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X POST
https://<IP>/redfish/v1/AccountService/Accounts -d '{ "UserName":"<USER>",
"Password":"<PASSWORD>", "RoleId":"<ROLE>", "Enabled":true}'
```

Example output:

```
{
"@Message.ExtendedInfo": [
{
"@odata.type": "#Message.v1_1_1.Message",
"Message": "The resource has been created successfully.",
"MessageArgs": [],
"MessageId": "Base.1.15.0.Created",
"MessageSeverity": "OK",
"Resolution": "None."
}
]
}
```

Delete User

Delete user form the system:

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X DELETE
https://<IP>/redfish/v1/AccountService/Accounts/<USER>
```

Example output:

```
{
"@Message.ExtendedInfo": [
{
"@odata.type": "#Message.v1_1_1.Message",
"Message": "The account was successfully removed.",
"MessageArgs": [],
"MessageId": "Base.1.15.0.AccountRemoved",
"MessageSeverity": "OK",
"Resolution": "No resolution is required."
}
]
}
```

User Management IPMI Commands

#	Function	Command

1	List the users	`ipmitool user list [<channel-number>]`
		For example:
		`ipmitool user list 1`

2	User creation	`ipmitool user set name <user-id> <username>`
		For example:
		`ipmitool user set name 2 Admin`

3	Set user password	`ipmitool user set password <user-id> <password>`
		For example:
		`ipmitool user set password 2 AdminPass_123`

4	Enable user	`ipmitool user enable <user-id>`
		For example:
		`ipmitool user enable 2`

5	Disable user	`ipmitool user disable <user-id>` For example:

#	Function	Command
		`ipmitool user disable 2`

6	Set user privilege	`ipmitool user priv <user-id> <privilege level(1-4)> [<channel-number>]`
		Where "privilege level":
		• 1 – callback level (currently not supported)
		• 2 – user level
		• 3 – operator level
		• 4 – administrator level
		For example:
		`ipmitool user priv 2 0x3 1`

7	Enable remote IPMI command functionality for user	`ipmitool channel setaccess [<channel-number>] <user id> ipmi = on| off`
		For example:
		`ipmitool channel setaccess 1 2 ipmi=on`

8	Lanplus commands to execute IPMI commands remotely for users with admin permissions	`ipmitool -C 17 -I lanplus -U <user> -P <password> -H <bmc-ip-address> <ipmi-command>`
		For example:
		`ipmitool -C 17 -I lanplus -U ADMIN -P AdminPass_123! -H 10.10.10.10 user list 1`

9	Lanplus commands to execute IPMI commands remotely for users with other than administrator roles	`ipmitool -C 17 -I lanplus -U <user> -P <password> -H <bmc-ip-address> -L <privilege (operator|user)> <ipmi-command>`

#	Function	Command

		For example:
		`ipmitool -C 17 -I lanplus -U operator1 -P operator123 -H 10.10.10.10 -L operator user list 1`
		`ipmitool -C 17 -I lanplus -U user1 -P user123 -H 10.10.10.10 -L user chassis status`

10	Delete user	`ipmitool user set name <user-id> ""`
		For example:
		`ipmitool user set name 2 ""`

Network Protocol Support

> ⚠ **Warning**
>
> To obtain the BMC's MAC address, refer to the DPU's board label.

BMC management network interface can be configured using Redfish or IPMI. By default, BMC comes up with the DHCP network configuration.

Network configuration functions:

- Setting DHCP/Static network mode configuration
- Adding/setting IPv4/IPv6 configuration including IP address, gateway, netmask
- Adding DNS servers
- Adding NTP server

- Setting BMC time with NTP server or system RTC

Network Management Redfish Commands

Get Network Protocol Configuration

```
curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/NetworkProtocol
```

Get Interface Configuration

```
curl -k -u root:'<password>' -XGET
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0
```

Enable/Disable Interface

```
curl -k -u root:'<password>' -XPATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0 -d
'{"InterfaceEnabled": <state>}'
```

Where `<state>` can be `true` or `false`.

Static IPv4 Address Configuration

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0 -d
'{"IPv4StaticAddresses": [{"Address": "<ip_addr>","SubnetMask": "
<netmask>","Gateway":"<gw_ip_addr>"}]}'
```

Example:

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0 -d
'{"IPv4StaticAddresses": [{"Address": "10.7.7.7","SubnetMask":
"255.255.0.0","Gateway":"10.7.0.1"}]}'
```

IPv4 DHCP Enable/Disable Configuration

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0 -d
'{"DHCPv4": {"DHCPEnabled": <state>}}'
```

Where `<state>` can be `true` or `false`.

Static DNS server IPv4 and IPv6 Configuration

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0 -d
'{"StaticNameServers": ["<dns_ip>"]}'
```

Static IPv6 Address Configuration

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0 -d
'{"IPv6StaticAddresses": [{"Address": "<ip>", "PrefixLength": <len>}]}'
```

Example:

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0 -d
'{"IPv6StaticAddresses": [{"Address": "fe80::3eec:efff:fe3b:e02f", "PrefixLength":
64}]}'
```

IPv6 DHCP Enable/Disable Configuration

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0 -d
'{"DHCPv6": {"OperatingMode": "<state>"}}'
```

Where `<state>` can be:

- `stateful` – DHCPv6 stateful mode is used to configure addresses, and when it is enabled, stateless mode is also implicitly enabled.
- `stateless` – DHCPv6 stateless mode allows configuring the interface using DHCP options but does not configure addresses. It is always enabled by default whenever DHCPv6 stateful mode is also enabled.
- `disabled` – DHCPv6 is disabled for this interface.

Enable NTP Configuration

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/NetworkProtocol -d '{"NTP":
{"ProtocolEnabled": <state>}}'
```

Where `<state>` can be `true` or `false`.

Static NTP Server IP Configuration

```
curl -k -u root:'<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/NetworkProtocol -d '{"NTP":
{"NTPServers": ["<ntp_server_ip>"]}}'
```

Network Management IPMI Commands

The following table lists the available network IPMI commands:

No.	Function	Command	Description

1	Change mode to Static	`ipmitool lan set 1 ipsrc <mode>`
		For example:
		`ipmitool lan set 1 ipsrc static`
			Sets LAN channel 1 IP config mode to static which corresponds to network interface "eth0"

2	Change mode to DHCP	`ipmitool lan set 1 ipsrc <mode>`
		For example:
		`ipmitool lan set 1 ipsrc dhcp`
			Sets LAN channel 1 IP config mode to DHCP which corresponds to the network interface "eth0"

3	Add IPv4 address	`ipmitool lan set 1 ipaddr <ip-address>`
		`ipmitool lan set 1 defgw ipaddr <ip-address>`
		`ipmitool lan set 1 netmask <netmask>`
			Adds IPv4 address, default gateway, and netmask to the network interface "eth0"

4	Get IPv4 config	`ipmitool lan print 1`	Gets IPv4 network config for channel 1 which corresponds to the network interface "eth0"

No.	Function	Command	Description
5	Set IPv6 address	`ipmitool lan6 set 1 nolock static_addr 0 enable <ipv6-address> 64`	Adds IPv6 address to the network interface "eth0"
6	Get IPv6 config	`ipmitool lan6 print 1`	Gets IPv6 network config for channel 1 which corresponds to the network interface "eth0"

7	Get DNS server	`ipmitool raw 0x32 0x6B`
		Output:
		`0b 31 30 2e 31 35 2e 31 32 2e 36 37`
		Corresponds to: `10.15.12.67`
			Gets the DNS server

8	Add DNS server	`ipmitool raw 0x32 0x6C 0x0b 0x31 0x30 0x2e 0x31 0x35 0x2e 0x31 0x32 0x2e 0x36 0x37`
		Output:
		`0x0b 0x31 0x30 0x2e 0x31 0x35 0x2e 0x31 0x32 0x2e 0x36 0x37`
		Corresponds to: `10.15.12.67`
			Adds the DNS server

9	Get NTP server	`ipmitool raw 0x32 0xA7`
		Output:
		`01 11 31 2e 69 6e 2e 70 6f 6f 6c 2e 6e 74 70 2e 6f 72 67`
		Where:
		• `01` – NTP status enable/disable
		• `11` – NTP server length
			Gets NTP server

No.	Function	Command	Description
		- `31 2e 69 6e 2e 70 6f 6f 6c 2e 6e 74 70 2e 6f 72 67` – NTP server address byte stream corresponds to `1.in.pool.ntp.org`	
10	Add NTP server	`ipmitool raw 0x32 0xA8 0x01 0x31 0x2e 0x69 0x6e 0x2e 0x70 0x6f 0x6f 0x6c 0x2e 0x6e 0x74 0x70 0x2e 0x6f 0x72 0x67` Where: - `31 2e 69 6e 2e 70 6f 6f 6c 2e 6e 74 70 2e 6f 72 67` – NTP server address byte stream corresponds to `1.in.pool.ntp.org`	Adds NTP server
11	Enable time sync to NTP server	`ipmitool raw 0x32 0xA8 0x02 0x01` Where: - `0x01` – enable NTP	Enables NTP time sync
12	Enable time sync to system RTC	`ipmitool raw 0x32 0xA8 0x02 0x00` Where: - `0x00` – disable NTP	Disables NTP time sync

Reset or Reboot BMC

Reboot BMC Redfish Command

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST -d
'{"ResetType": "GracefulRestart"}'
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Actions/Manager.Reset
```

Reboot BMC IPMI Command

```
ipmitool mc re cold
```

Factory Reset BMC

The following commands factory reset the BMC configuration.

Factory Reset Redfish Command

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

> ⓘ **Important**
>
> Before connecting to the internet, it is important to change the default global password to prevent potential malicious attackers from hacking your system. For information on password policy, refer to section "BMC Management Interface".

Factory Reset IPMI Command

```
ipmitool raw 0x32 0x66
```

After issuing the ipmitool raw command for factory reset, you must log into the BMC and reboot it for the factory reset to take effect.

⚠ **Warning**

If you have lost your BMC login credentials and cannot login, you may issue the following command from the BlueField Arm:

```
ipmitool mc reset cold
```

ⓘ **Important**

Before connecting to the internet, it is important to change the default global password to prevent potential malicious attackers from hacking your system. For information on password policy, refer to section "BMC Management Interface".

BMC and CEC Firmware Update

Firmware upgrade of BMC and CEC components using BMC can be performed from a remote server using the Redfish interface.

No.	Function	Command	Required for BMC/CEC Update	Description
1	Establish Redfish connection session	``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 '"'` `` Where: - bmc_ip – BMC IP address - password – password of root user	BMC CEC	Establish Redfish connection session
2	Trigger a secure firmware update	`curl -k -u root:'<password>' -H "Content-Type: application/octet-stream" -X POST -T <package_path> https://<bmc_ip>/redfish/v1/UpdateService/update` Where: - password – password of root user - bmc_ip – BMC IP address - package_path – firmware update package path	BMC CEC	Triggers the secure update and starts tracking the secure update progress

3	Track secure firmware update progress	`curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/TaskService/Tasks`
		Find the current task ID in the response and use it for checking the progress:
		`curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/TaskServic`
			BMC
			CEC
				Tracks the firmware update progress

No.	Function	Command	Required for BMC/CEC Update	Description
		`e/Tasks/<task_id> | jq -r ' .PercentComplete'` Where: • password – password of root user • bmc_ip – BMC IP address • task_id – Task ID		
4	Reset/reboot a BMC	`curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST -d '{"ResetType": "GracefulRestart"}' https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Actions/Manager.Reset` Where: • password – password of root user • bmc_ip – BMC IP address	BMC	Resets/reboots the BMC
5	Fetch running BMC firmware version	For BlueField-3: `curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory/BMC_Firmware | jq -r ' .Version'` Where: • password – password of root user • bmc_ip – BMC IP address	BMC	Fetches the running firmware version from BMC
		For BlueField-2:		

No.	Function	Command	Required for BMC/CEC Update	Description

		`curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory`
		Fetch the current firmware ID and then perform:
		`curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory/<firmware_id>_BMC_Firmware | jq -r ' .Version'`
		Where:
		• `password` – password of root user
		• `bmc_ip` – BMC IP address
		• `firmware_id` – numeric value found in the `FwInventory` schema only. It is calculated during firmware update by the BMC and used to distinguish between the versions.		

6	Fetch running CEC firmware version	`curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory/Bluefield_FW_ERoT | jq -r ' .Version'` Where: • `password` – password of root user • `bmc_ip` – BMC IP address	CEC	Fetches the running firmware version from CEC

BMC Update

Note

Firmware update takes about 12 minutes.

After initiating the BMC secure update with the command #2 to from the previous table, a response similar to the following is received:

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

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

Command #3 from the previous table can be used to track secure firmware update progress. For instance:

```
curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/TaskService/Tasks/0
| jq -r ' .PercentComplete'

% 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
```

Command #3 is used to verify the task has completed because during the update procedure the reboot option is disabled. When "PercentComplete" reaches 100, command #4 is used to reboot the BMC. For example:

```
curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/TaskService/Tasks/0
| jq -r ' .PercentComplete'
% Total % Received % Xferd Average Speed Time Time Time Current Dload Upload
Total Spent Left Speed
100 3822 100 3822 0 0 81319 0 --:--:-- --:--:-- --:--:-- 81319
100

curl -k -u root:'<password>' -H "Content-Type: application/octet-stream" -X POST -d
'{"ResetType": "GracefulRestart"}'
https://<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"
}
]
}
```

Command #5 can be used to verify the current BMC firmware version after reboot:

- For BlueField-3:

  ```
  curl -k -u root:'<password>' -X GET
  https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory/BMC_Firmware
  | jq -r ' .Version'

  % Total % Received % Xferd Average Speed Time Time Time Current Dload
  Upload Total Spent Left Speed
  100 513 100 513 0 0 9679 0 --:--:-- --:--:-- --:--:-- 9679
  ```

- For BlueField-2:
    1. Fetch the firmware ID from `FirmwareInventory`:

       ```
       curl -k -u root:'<password>' -X GET
       https:/<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory/
       {
       "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory",
       "@odata.type":
       "#SoftwareInventoryCollection.SoftwareInventoryCollection",
       "Members": [
       {
       "@odata.id":
       "/redfish/v1/UpdateService/FirmwareInventory/8c8549f3_BMC_Firmware"
       ...
       ```

    2. Use command #5 with the fetched firmware ID in the previous step:

       ```
       curl -k -u root:'<password>' -X GET
       https:/<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory/8c8549f3_BM
       | jq -r ' .Version'

       % Total % Received % Xferd Average Speed Time Time Time Current
       Dload Upload Total Spent Left Speed
       100 471 100 471 0 0 622 0 --:--:-- --:--:-- --:--:-- 621
       bmc-23.04
       ```

CEC Update

ⓘ **Note**

Firmware update takes about 20 seconds.

After initiating the BMC secure update with the command #2 to from the previous table, a response similar to the following is received:

```
curl -k -u root:'<password>' -H "Content-Type: application/octet-stream" -X POST -T
<package_path> https://<bmc_ip>/redfish/v1/UpdateService
{
"@odata.id": "/redfish/v1/TaskService/Tasks/0",
"@odata.type": "#Task.v1_4_3.Task",
"Id": "0",
"TaskState": "Running"
}
```

Command #3 can be used to track the progress of the CEC firmware update. For example:

```
curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/TaskService/Tasks/0
| jq -r ' .PercentComplete'
% Total % Received % Xferd Average Speed Time Time Time Current Dload Upload
Total Spent Left Speed
100 2123 100 2123 0 0 38600 0 --:--:-- --:--:-- --:--:-- 37910
100
```

After the CEC secure update operation is complete, a power cycle or cold reset of the BlueField-3 DPU must be manually triggered to apply the changes once the update is finished.

Command #6 can be used to verify the current CEC firmware version after reboot:

```
curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory/Bluefield_FW_ERoT
| jq -r ' .Version'
```

```
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 421 100 421 0 0 1172 0 --:--:-- --:--:-- --:--:-- 1172
19-4
```

CEC Activation and Reset

> ⚠ **Warning**
>
> This is relevant only for BlueField-3 DPUs only.

To activate the new CEC firmware, it is necessary to reset the CEC device. The following options are available:

- Reset the entire BlueField DPU, which typically involves a full power cycle of the host platform.
- Reset the CEC and BMC subsystems only. This can be done using the ipmitool i2c command over the SMBus channel connected to the PCIe golden finger.

  > ⚠ **Warning**
  >
  > This option is valid only for servers which support I2C over SMBus from the host BMC.

These options provide flexibility in managing the CEC device to apply the firmware update as needed.

To trigger the CEC reset:

```
ipmitool raw 0x06 0x52 <BUS-ID> 0x82 0x00 0x03 0xFE
ipmitool raw 0x06 0x52 <BUS-ID> 0x82 0x00 0x01 0xFE
sleep <100ms>
ipmitool raw 0x06 0x52 <BUS-ID> 0x82 0x00 0x01 0xFF
ipmitool raw 0x06 0x52 <BUS-ID> 0x82 0x00 0x03 0xFF
```

⚠ **Warning**

The `BUS-ID` value is system related. It relays how the host BMC is connected to the SMBus of the related DPU.

⚠ **Warning**

The format of the `ipmitool i2c` command is as follows:

```
ipmitool raw <netfun> <cmd> <bus-id> <addr> <read-count>
<write-data1> <write-data2>
```

CEC Background Update Status

ⓘ **Note**

This section is relevant only for BlueField-3.

BMC and CEC have an active and inactive copy of the same firmware image on their respective firmware SPI flash. The firmware update updates the inactive copy, and on a successful boot from the newly updated and active image, the inactive image (e.g., the previous active image) is updated with the latest image.

Warning

Firmware update cannot be initiated if the background copy is in progress.

To check the status of the background update:

```
curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/Chassis/Bluefield_ERoT
...
"Oem": {
"Nvidia": {
"@odata.type": "#NvidiaChassis.v1_0_0.NvidiaChassis",
"AutomaticBackgroundCopyEnabled": true,
"BackgroundCopyStatus": "Completed",
"InbandUpdatePolicyEnabled": true
}
}
...
```

Note

The background update initially indicates `InProgress` while the inactive copy of the image is being updated with the copy.

Possible Error Codes

> **Note**
>
> This section is relevant only for BlueField-3.

Fault	Diagnosis and Possible Solution
Connection to BMC breaks during firmware package transfer	• 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 A new firmware update can be attempted by the Redfish client.
Connection to BMC breaks during firmware update	• 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: ◦ 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 A new firmware update can be attempted by the Redfish client.
Two firmware update requests are initiated	The Redfish server blocks the second firmware update request and returns the following: • HTTP code 400 "Bad Request" • Redfish message based on standard registry entry `UpdateInProgress` Check the status of the ongoing firmware update by looking at the `TaskCollection` resource.
Redfish task hangs	• Redfish task URI that previously returned by the Redfish server is no longer accessible • PLDM-based firmware update progresses

Fault	Diagnosis and Possible Solution
	- After a reboot of BMC, or restart/recovery of the Redfish server, the Redfish server us in `idle` state A new firmware update can be attempted by the Redfish client.
BMC-EROT communication failure during image transfer	The Redfish task monitoring the firmware update indicates a failure: - `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 The Redfish client may retry the firmware update.
Firmware update fails	The Redfish task monitoring the firmware update indicates a failure: - `TaskState` is set to `Exception` - `TaskStatus` is set to `Warning` - `Messages` array in the task includes an entry describing the error The Redfish client may retry the firmware update.
ERoT failure (not responding)	The Redfish task monitoring the firmware update indicates a failure: - `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 The Redfish client may retry the firmware update.
Firmware image validation failure	The Redfish task monitoring the firmware update indicates a failure: - `TaskState` is set to `Exception` - `TaskStatus` is set to `Warning`

Fault	Diagnosis and Possible Solution
	- 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 The Redfish client might retry the firmware update.
Power loss before activation command is sent	- The Redfish server is in idle state A new firmware update can be attempted by the Redfish client.
Firmware activation failure	The Redfish task monitoring the firmware update indicates a failure: - 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 The Redfish client may retry the firmware update.
Push to BMC firmware package greater than 200 MB	- 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.

BlueField BMC Redfish Triggers

Redfish triggers allow the user to get a journal message when a certain metric crosses a defined threshold for a defined time:

- The trigger threshold can only be a numeric threshold
- The trigger thresholds are unrelated to the sensor thresholds
- The maximum number of triggers allowed in the system is 10

For more details, refer to Redfish Resource and Schema Guide.

No.	Function	Command	Description
1	Add a numeric trigger	`curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST https://<bmc_ip>/redfish/v1/TelemetryService/Triggers/ -d '{"Id":"< >","Name":"<>","MetricType":"<>","TriggerActions":["<>"],"NumericThresholds":{"<>":{"Activation":"<>","DwellTime":"<>","Reading":<>}},"MetricProperties":["<>"]}'`	Adds a numeric trigger to the BMC
2	Delete a trigger	`curl -k -u root:'<password>' -H "Content-Type: application/json" -X DELETE https://<bmc_ip>/redfish/v1/TelemetryService/Triggers/<trigger-name>`	Deletes a trigger

Redfish Certificate Management

Certificate management actions (e.g., getting certificate information, doing atomic replacement of certificates) are found in the CertificateService resource.

The CertificateLocations resource is responsible for providing inventory of all the certificates which the service manages.

More details can be found in the Redfish Certificate Management White Paper.

No.	Function	Command	Description
1	Get certificate locations	`curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/CertificateService/CertificateLocations`	Inventory of all certificates the service is managing
2	Get certificate	`curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/Managers/Bluefield_BM`	Get certificate info

No.	Function	Command	Description
	Information	`C/NetworkProtocol/HTTPS/Certificates/1`	
3	Replace existing certificate	`curl -k -u root:'<password>' -X POST https://<bmc_ip>/redfish/v1/CertificateService/Actions/CertificateService.ReplaceCertificate -d @certificate.json`	Replace certificate
4	Generate CSR	`curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST https://<bmc_ip>/redfish/v1/CertificateService/Actions/CertificateService.GenerateCSR -d @csr_file.json`	Generate certificate signing request
5	Install a certificate	`curl -k -u root:'<password>' -H "Content-Type: application/octet-stream" -X POST https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/NetworkProtocol/HTTPS/Certificates -d @certificate.json`	Install a certificate

NIC Subsystem Management

⚠ **Warning**

This content is relevant for BlueField-3 devices only.

Redfish NIC Subsystem Management

Get Operation Mode

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

ⓘ **Note**

See status under "Mode".

Change to DPU Mode

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST -d
'{"Mode":"DpuMode"}'
```

```
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Oem/Nvidia/Actions/Mode.Set
```

Change to NIC Mode

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST -d
'{"Mode":"NicMode"}'
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Oem/Nvidia/Actions/Mode.Set
```

IPMItool NIC Subsystem Management

Since the standard IPMItool commands do not cover all functionality, a set of custom NVIDIA IPMItool raw commands is available to enable configuring the NIC subsystem on the DPU directly.

IPMItool raw commands follow the following format:

```
ipmitool -C 17 -I lanplus -H <bmc_ip_addr> -U <username> -P <password> raw
<netfunc> <cmd> <data>
```

Where:

- netfunc – network function which identifies the functional message class, and clusters IPMI commands into sets
- cmd – one byte command within a network function
- data – optional element which provides additional parameters for a request or response message

The following table lists the supported IPMItool raw commands:

<table>
<tr><th>netfunc</th><th>cmd</th><th>data</th><th>Description</th></tr>
<tr><td>0x32</td><td>0x9A</td><td>N/A</td><td>Get external host privileges.
Prints current state for all fields:
<table>
<tr><th>Byte</th><th>Field</th></tr>
<tr><td>0</td><td>HOST_PRIV_FLASH_ACCESS</td></tr>
<tr><td>1</td><td>HOST_PRIV_FW_UPDATE</td></tr>
<tr><td>2</td><td>HOST_PRIV_NIC_RESET</td></tr>
<tr><td>3</td><td>HOST_PRIV_NV_GLOBAL</td></tr>
<tr><td>4</td><td>HOST_PRIV_NV_HOST</td></tr>
<tr><td>5</td><td>HOST_PRIV_NV_INTERNAL_CPU</td></tr>
<tr><td>6</td><td>HOST_PRIV_NV_PORT</td></tr>
<tr><td>7</td><td>HOST_PRIV_PCC_UPDATE</td></tr>
</table>
Each state is represented by binary byte in order.
<ul>
<li>00 – Default</li>
<li>01 – Enabled</li>
<li>02 – Disabled</li>
</ul></td></tr>
<tr><td>0x32</td><td>0x9B</td><td>Byte0
Byte1</td><td>Set external host privilege.
Byte0 selects privilege according to the following table:
<table>
<tr><th>Byte</th><th>Field</th></tr>
<tr><td>0</td><td>HOST_PRIV_FLASH_ACCESS</td></tr>
<tr><td>1</td><td>HOST_PRIV_FW_UPDATE</td></tr>
<tr><td>2</td><td>HOST_PRIV_NIC_RESET</td></tr>
<tr><td>3</td><td>HOST_PRIV_NV_GLOBAL</td></tr>
<tr><td>4</td><td>HOST_PRIV_NV_HOST</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>netfunc</th><th>cmd</th><th>data</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Byte</th><th>Field</th></tr>
<tr><td>5</td><td>HOST_PRIV_NV_INTERNAL_CPU</td></tr>
<tr><td>6</td><td>HOST_PRIV_NV_PORT</td></tr>
<tr><td>7</td><td>HOST_PRIV_PCC_UPDATE</td></tr>
</table>
Byte1 is the value being set.
Supported values:
<ul>
<li>00 – Default</li>
<li>01 – Enabled</li>
<li>02 – Disabled</li>
</ul>
<b>Note</b>
<ul>
<li>Currently, firmware does not support the parameters HOST_PRIV_FLASH_ACCESS and HOST_PRIV_PCC_UPDATE. Their value should stay as DEVICE_DEFAULT.</li>
<li>The parameter HOST_PRIV_NV_INTERNAL_CPU should either equal the parameter HOST_PRIV_NV_GLOBAL or one of them should be set to DEVICE_DEFAULT.</li>
<li>If the parameter HOST_PRIV_FLASH_ACCESS is not set to DEVICE_DEFAULT then the following parameters should all be set to DEVICE_DEFAULT or be equal to the value of HOST_PRIV_FLASH_ACCESS: HOST_PRIV_NV_HOST, HOST_PRIV_NV_PORT, HOST_PRIV_NV_GLOBAL, HOST_PRIV_NV_INTERNAL_CPU, HOST_PRIV_PCC_UPDATE, HOST_PRIV_FW_UPDATE.</li>
</ul>
</td></tr>
</table>

<table>
<tr><th>netfunc</th><th>cmd</th><th>data</th><th>Description</th></tr>
<tr><td>0x32</td><td>0x9C</td><td>N/A</td><td>Get SmartNIC mode. Prints current configuration: INTERNAL_CPU_OFFLOAD_ENGINE.
- 00 – Disabled
- 01 – Enabled</td></tr>
<tr><td>0x32</td><td>0x9D</td><td>Byte0</td><td>Set SmartNIC mode (INTERNAL_CPU_OFFLOAD_ENGINE) to Byte0.
Supported values:
- 00 – Disabled
- 01 – Enabled</td></tr>
<tr><td>0x32</td><td>0x9E</td><td>N/A</td><td>Get host access. Prints current HOST_PRIV_RSHIM.
- 00 – Disabled
- 01 – Enabled</td></tr>
<tr><td>0x32</td><td>0x9F</td><td>Byte0</td><td>Set host access. Sets HOST_PRIV_RSHIM to Byte0.
Supported values:
- 00 – Disabled
- 01 – Enabled</td></tr>
<tr><td>0x32</td><td>0xA2</td><td>N/A</td><td>Query strap options. Prints current state for all fields:
<table>
<tr><th>Byte</th><th>Field</th></tr>
<tr><td>0</td><td>VERSION</td></tr>
<tr><td>1</td><td>DISABLE_INBAND_RECOVER_VALUE</td></tr>
<tr><td>2</td><td>PRIMARY_IS_PCORE_1_VALUE</td></tr>
<tr><td>3</td><td>2PCORE_ACTIVE_VALUE</td></tr>
<tr><td>4</td><td>SOCKET_DIRECT_VALUE</td></tr>
<tr><td>5</td><td>PCI_REVERSAL_VALUE</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>netfunc</th><th>cmd</th><th>data</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Byte</th><th>Field</th></tr>
<tr><td>6</td><td>PCI_PARTITION_1_VALUE</td></tr>
<tr><td>7</td><td>PCI_PARTITION_0_VALUE</td></tr>
<tr><td>8</td><td>OSC_FREQ_1_VALUE</td></tr>
<tr><td>9</td><td>OSC_FREQ_0_VALUE</td></tr>
<tr><td>10</td><td>CORE_BYPASS_N_VALUE</td></tr>
<tr><td>11</td><td>FNP_VALUE</td></tr>
<tr><td>12</td><td>DISABLE_INBAND_RECOVER_VALUE</td></tr>
<tr><td>13</td><td>PRIMARY_IS_PCORE_1_MASK</td></tr>
<tr><td>14</td><td>2PCORE_ACTIVE_MASK</td></tr>
<tr><td>15</td><td>SOCKET_DIRECT_MASK</td></tr>
<tr><td>16</td><td>PCI_REVERSAL_MASK</td></tr>
<tr><td>17</td><td>PCI_PARTITION_1_MASK</td></tr>
<tr><td>18</td><td>PCI_PARTITION_0_MASK</td></tr>
<tr><td>19</td><td>OSC_FREQ_1_MASK</td></tr>
<tr><td>20</td><td>OSC_FREQ_0_MASK</td></tr>
<tr><td>21</td><td>CORE_BYPASS_N_MASK</td></tr>
<tr><td>22</td><td>FNP_MASK</td></tr>
</table>
Each state is represented by binary byte in order.
Supported values:
• 00 – Disabled
• 01 – Enabled</td></tr>
<tr><td>0x32</td><td>0xA3</td><td>N/A</td><td>Get SmartNIC OS State.</td></tr>
</table>

netfunc	cmd	data	Description
			• 00 – BootRom • 01 – BL2 • 02 – BL31 • 03 – UEFI • 04 – OsStarting • 05 – OsIsRunning • 06 – LowPowerStandby • 07 – FirmwareUpdateInProgress • 08 – OsCrashDumpInProgress • 09 – OsCrashDumpIsComplete • 0A – FWFaultCrashDumpInProgress • 0B – FWFaultCrashDumpIsComplete • 0C – Invalid

Changing Operation Mode

netfunc	cmd	data	Description
0x32	0x9D	0x1	Change to DPU mode
0x32	0x9D	0x0	Change to NIC mode

Enable/Disable RShim from Host

netfunc	cmd	data	Description
0x32	0x9F	0x1	Enable RShim from host
0x32	0x9F	0x0	Disable RShim from host

NVIDIA OEM Commands

Not all functionalities are covered with a standard set of IPMItool commands. Therefore, a set of custom NVIDIA IPMItool raw commands have been added. The first two parameters of the raw command are NetFN and CMD.

IPMItool raw commands follow the following format:

```
ipmitool -C 17 -I lanplus -H <bmc_ip> -U <username> -P <password> raw <netfunc> <cmd> <data>
```

Where:

- netfunc – network function which identifies the functional message class, and clusters IPMI commands into sets
- cmd – one byte command within a network function
- data – optional element which provides additional parameters for a request or response message

netfunc	cmd	data	Description
0x32	0x66	N/A	Factory reset
0x32	0x67	0x00	Disable vendor field mode settings to be run from Arm OS
0x32	0x67	0x01	Enable vendor field mode settings to be run from Arm OS
0x32	0x68	N/A	Fetch vendor field mode settings to be run from Arm OS
0x32	0x6a	0	Stops RShim on BMC

netfunc	cmd	data	Description
0x32	0x6a	1	Starts RShim on BMC
0x32	0x69	N/A	Retrieves RShim service status on BMC. Expected output: • 0x00 – RShim inactive (default state) • 0x01 – RShim active
0x32	0x6b	N/A	Gets the DNS server
0x32	0x6c	0x0b 0x31 0x30 0x2e 0x31 0x35 0x2e 0x31 0x32 0x2e 0x36 0x37	Adds the DNS server
0x32	0x92	N/A	Enters the DPU into Livefish (FNP) mode
0x32	0x93	N/A	Disable Livefish (FNP) mode
0x32	0xa1	0x0	OEM command 0xa1 is defined for various reset controls of NVIDIA® BlueField® from BMC under the OEM NetFn group 0x30. • 0x00 – hard reset of BlueField DPU
0x32	0xa7	N/A	Gets NTP server
0x32	0xa8	0x01 0x31 0x2e 0x69 0x6e 0x2e 0x70 0x6f 0x6f 0x6c 0x2e 0x6e 0x74 0x70 0x2e 0x6f 0x72 0x67	Adds NTP server
0x32	0xa8	0x02 0x01	Enable time sync to NTP server
0x32	0xa8	0x02 0x00	Disables NTP time sync

Table of Common Commands

Capability	Redfish	IPMItool
Changing the default BMC password	Changing default password using Redfish	N/A
Changing the default UEFI password	Changing UEFI Password	N/A
Enabling/disabling secure boot	Setting Secure Boot State	N/A
Updating BMC firmware	BMC and CEC firmware update	N/A
Updating DPU BFB	Pushing BFB from BMC to BlueField Arm	N/A
Configuring DPU to network boot from the out-of-band interface first	Boot Config Using Redfish	Boot Config Using IPMI
Resetting DPU		Reset control
Resetting DPU BMC	Reset control using Redfish	Reset control using IPMI
Factory reset	Factory Reset Redfish Command	Factory Reset IPMI Command
Getting DPU versions	System inventory	N/A
Getting DPU BMC versions	Retrieving BMC version using Redfish	Retrieving BMC version using IPMI command
Getting high-speed ports MAC addresses	Chassis Card1	List of IPMI

Capability	Redfish	IPMItool
for mapping DPUs' Ethernet devices	NetworkAdapters	Supported FRUs
DPU monitoring (SEL, FRU, etc.)		
User management	User management Redfish commands	User management IPMI commands
Enabling secure boot with customer keys	BIOS secure boot configuration	N/A
Enabling/disabling zero-trust mode	N/A	Enable/disable RShim from Host
Enabling RShim from DPU BMC	Enable RShim on DPU BMC	Enable RShim
Changing DPU mode	Redfish NIC Subsystem Management	Changing operation mode
Partial BFB update (ATF/UEFI)		
Updating BFB using simple update and "MultipartHttpPushUri"		

List of Supported IPMItool Commands

The IPMItool program allows you to remotely manage the IPMI functions of the NVIDIA® BlueField® BMC. The commands below may be directed to the BMC's Ethernet interface by invoking:

```
ipmitool -C 17 -I lanplus -H <bmc_ip_addr> -U ADMIN -P ADMIN
<ipmitool_arguments>
```

The following list provides a full list of the IPMItool arguments supported by BlueField BMC.

```
chassis power reset
chassis status (to be implemented in future release)
fru
fru print 0
fru print 1
fru read 0 /tmp/fru
fru read 1 /tmp/fru
lan print
mc info
mc reset cold
sdr elist
sdr get <sensor name>
sdr list
sdr type <type>
sel
sel clear
sel elist
sel listsensor get <sensor name>
```

```
sensor list
sol activate
user disable <user id>
user enable <user id>
user list [<channel number>]
user priv <user id> <privilege level(1-4)> [<channel number>]
user set name <user id> <user name>
user set password <user id> <password>
```

Appendix – Software Upgrade Provisioning Flow

This appendix details the steps for provisioning software components on the NVIDIA® BlueField®-3 DPU.

Note

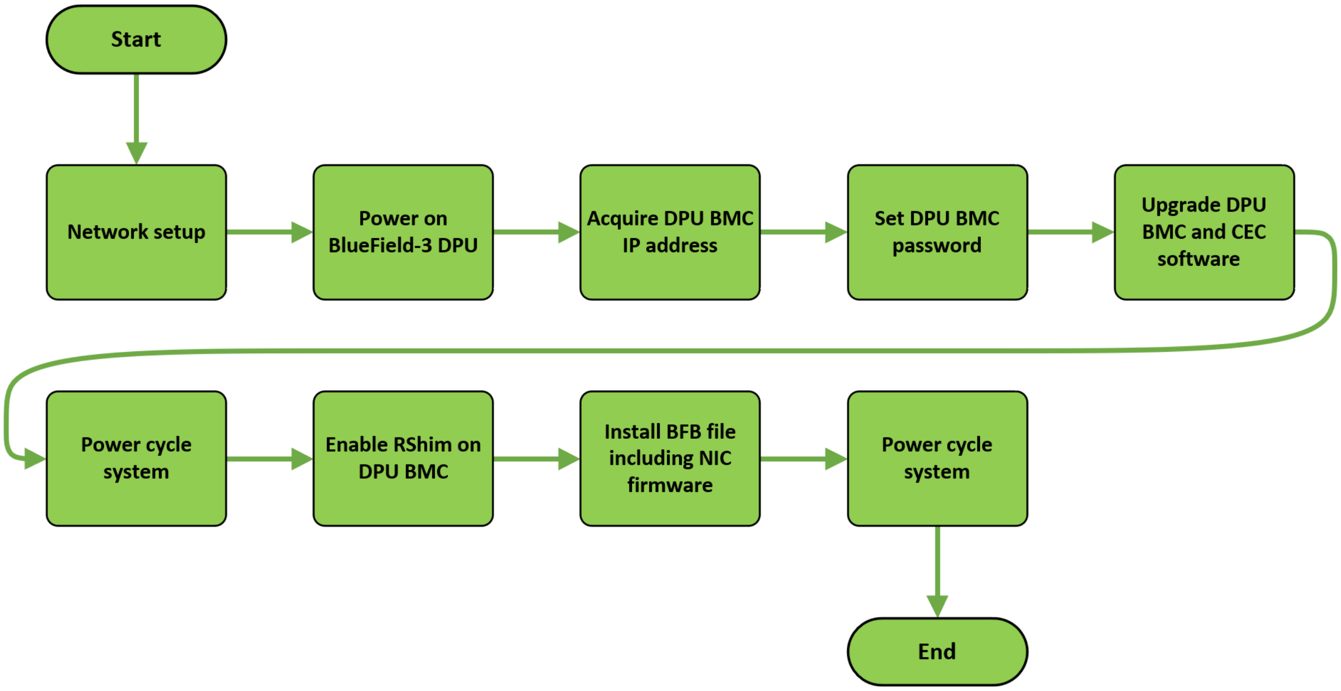
The procedure for DPU BMC software upgrade is agnostic to the version of the software. Once upgraded, however, the procedure assumes you to be running the latest BMC software.

This workflow guarantees the most current software to be installed on various components of the BlueField-3 DPU. This includes:

- DPU BMC
- CEC
- Arm ATF
- Arm UEFI
- Arm OS
- NIC firmware

The process aims to ensure that all these components are up to date.

The following high-level flow diagram outlines the expected steps to be followed throughout the process:



1. Establish a connection between the onboard RJ-45 network interface and the management network. Refer to section "Network Protocol Support" for detailed instructions on network connectivity.

2. Power on the BlueField DPU. This can be accomplished manually or by utilizing either ipmitool or Redfish commands directed at the host's BMC.

    - IPMItool example:

    ```
    ipmitool -H <bmc_ip_or_hostname> -U <username> -P <password>
    power on
    ```

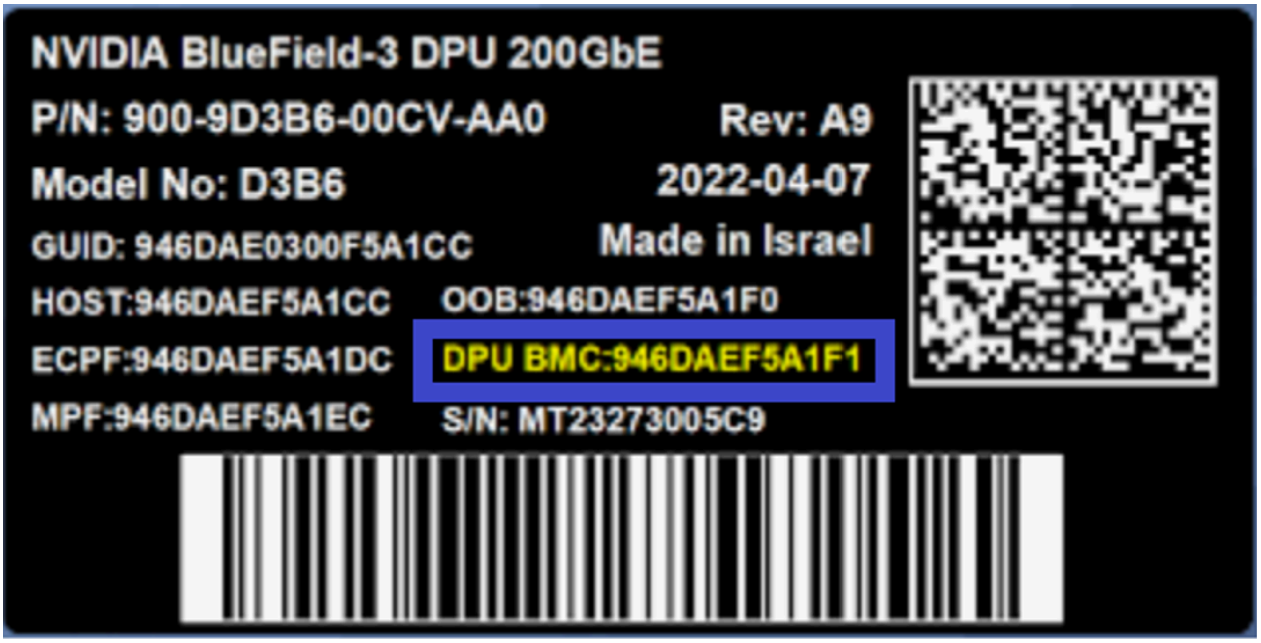
    Replace the parameters with the information relevant for your host BMC.

    - Redfish example:

    ```
    curl -X POST -k -u root:<password> -H "Content-Type: application/json" -d
    '{"ResetType": "On"}'
    https://<bmc_ip>/redfish/v1/Systems/<System_ID>/Actions/ComputerSyste
    ```

    Replace the parameters with the information relevant for your host BMC.

3. Acquire the DPU BMC's IP address from the label affixed to the DPU (highlighted in the image). Use the DPU BMC's MAC address to retrieve the assigned IP address from the DHCP server to enable communication with the DPU BMC over the network.



NVIDIA BlueField-3 DPU 200GbE
P/N: 900-9D3B6-00CV-AA0 Rev: A9
Model No: D3B6 2022-04-07
GUID: 946DAE0300F5A1CC Made in Israel
HOST:946DAEF5A1CC OOB:946DAEF5A1F0
ECPF:946DAEF5A1DC DPU BMC:946DAEF5A1F1
MPF:946DAEF5A1EC S/N: MT23273005C9

4. If the BlueField-3 DPU is a new device which has not yet been provisioned, the DPU BMC comes from the factory with a default password (0penBmc). To establish communication with the DPU BMC, you must change the default password. Refer to section "Changing Default Password" for instructions on changing the default password of the DPU-BMC.

5. Upgrade DPU BMC and CEC software. This step is crucial for guaranteeing that all new features and functionalities are available on your device. Refer to section "BMC and CEC Firmware Update" for instructions on how to do that.

6. Power cycle the host. This can be accomplished by utilizing either ipmitool or Redfish commands directed at the host's BMC:

    1. IPMItool example:

    ```
    ipmitool -H <bmc_ip_or_hostname> -U <username> -P <password> power cycle
    ```

    Replace the parameters with the information relevant for your host BMC.

    2. Redfish example:

       ```
       curl -k -u root:<password> -X POST
       "https://<host_bmc_ip>/redfish/v1/Systems/1/Actions/ComputerSystem.Res
       -d '{"ResetType": "ForceRestart"}'
       ```

       Replace the parameters with the information relevant for your host BMC.

7. Ensure that the RShim is disconnected from the host to enable the DPU BMC to take ownership of it. To achieve this, follow the following steps in section "Enabling RShim on BMC" under "Installing BFB".

8. Install the BFB file and NIC firmware.

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

   Follow the instructions provided in the BFB image transfer guidelines provided in section "Transferring BFB Image" under "Installing BFB" while utilizing the newly created BFB file, `new.bfb`.

9. To ensure that the new NIC firmware takes effect, perform a final power cycle of the system as detailed in step 6.

Document Revision History

Rev v23.10 – November 30, 2023

Added:

- Section "Power Capping"
- Section "DPU Chassis"
- Section "BlueField Console Log"
- Section "Viewing Currently Installed CA Certificates"
- Section "CA Certificates Collection Modification"
- Section "Enable RShim on DPU BMC"
- Section "Network Management Redfish Commands"
- Section "CEC Update" under "BMC and CEC Firmware Update"
- Section "OOB Network 3-Port Switch Control"
- Appendix "Provisioning Software Upgrade Flow"

Updated:

- Section "Boot Config Using Redfish"
- Section "Installing BFB"
- Section "Golden Images Reprovisioning"
- Section "Factory Reset BMC"
- Section "Reset or Reboot BMC"

- Section "BMC Sensor Data"
- Section "Retrieving Data from BlueField Via IPMB"
- Section "Retrieving Data from BMC Via IPMB"
- Section "Serial Over LAN (SOL)"
- Section "Expected Output"
- Section "BMC Dump Operations"

Rev v23.09 – September 20, 2023

Added:

- Section "System Inventory"
- Section "DPU Chassis"
- Section "NIC Subsystem Management"
- Section "Table of Common Commands"

Updated:

- Section "FRU Reading"
- Section "System Event Log"
- Section "List of IPMI Supported FRUs"
- Section "Boot Configuration"
- Section "2024-02-23_09-31-43_BIOS Secure Boot Configuration"
- Section "BIOS Configuration"
- Section "Reset Control"

Rev v23.07 – August 10, 2023

Added:

- Section "Changing Default Password"
- Section "Account Service"
- Section "Configuring BIOS Secure Boot"
- Section "Configuring BIOS"
- Section "Redfish Certificate Management"
- The commands `0x32 0x97` and `0x32 0x98` to "NVIDIA Custom Commands"

Updated:

- Note in section "Network Protocol Support"
- Section "Boot Order Config"
- Section "Installing BFB"
- Section "BMC and CEC Firmware Update" and its subsections

Rev v23.04 – May 17, 2023

Added:

- Figure "NVIDIA® BlueField®-3 BMC Connector" to section "BMC Console Interface"
- Section "SEL Messages"
- Section "Updating BMC and Glacier Firmware with Vendor Field Mode" which is relevant for NVIDIA® BlueField®-3 DPU only
- Page "Serial Redirect Mode"
- Section "BlueField BMC Redfish Triggers"
- Command `0x32 0x92` and `0x32 0x93` to "NVIDIA Custom Commands" table

Updated:

- Section "BMC Management Interface" with new password requirements
- Section "Sensor Data Record (SDR) Repository"
- Link status codes to the p0_link and p1_link sensors in section "List of IPMI Supported Sensors"
- Section "BMC and CEC Firmware Update"

Rev 2.8.2-34 – October 21, 2022

Added:

- Page "Vendor Field Mode"
- Section "DPU Reset"

Updated:

- Section "Boot Order Config" with note on DPU boot override setting

Rev 2.8.2 – June 01, 2022

Updated:

- NIC thermal sensors line in table under s ection "SDR Entry List"

Rev 2.8.2 – April 04, 2022

Updated:

- Page "NVIDIA OEM Commands"

Rev 2.8.2 – January 04, 2022

Added:

- New password policy to:
  - Warning box in section "BMC Management Interface"

    - Section "Boot Sequence Overview"
    - Step 3 in section "User Management"
- Section "Changing Default Credentials Using bf.cfg"
- `-C 17` argument to IPMItool `lanplus` commands

Updated:

- Section "User Management"
- Section "Reset Control"

Rev 2.8.2 – October 25, 2021

Updated:

- Section "Pushing Bootstream from BMC to BlueField-2 Arm"
- Section "BMC Management Interface" by removing mentions of interface `eth1`
- Page "NVIDIA OEM Commands"