NVIDIA BlueField BMC Software v26.04

Table of Contents

1 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® networking device (DPU or SuperNIC) and is intended for those who want to familiarize themselves with the functionality provided by the BMC.

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

If this is your first time deploying the BlueField device (Day 0), please refer to the "First-time Installation Procedure" page for a walkthrough of the provisioning process. The rest of this manual is provides information for BlueField lifecycle management (Day 1) activities.

1.1 Software Download

To download product software, please refer to the NVIDIA DOCA Software Framework page.

1.2 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.

1.3 Related Documentation

Document Name	Description
NVIDIA BlueField DPU BSP	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 device 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

Document Name	Description
NVIDIA BlueField-3 Ethernet DPU User Guide	This manual describes BlueField-3 Ethernet device 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®
Redfish Data Model Specification	This document includes the informative and normative descriptions copied from the description and long description annotations in the Redfish Schema Bundle (DSP8010), and adds supplemental normative text to further explain the usage of particular properties or resources.
IPMI Architecture GitHub	This document describes the architecture of IPMI design.

1.4 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
NIC	Network interface card
SoC	System-on-chip
SOL	Serial Over LAN
SEL	System Event Log
SDR	Sensor Data Record; Sensor Data Repository
UART	Universal Asynchronous Receiver Transmitter

2 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.

2.1 Changes and New Features

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

2.1.1 Changes and New Features in 26.04

- Aligned the BMC's `bluefield_temp` sensor thresholds with the official thermal limits of the NIC architecture. This update ensures accurate thermal monitoring and resolves previous reporting discrepancies. The new thresholds are:
  - Upper critical (UC) – 96°C (decreased from 105°C)
  - Upper non-critical (UNC) – 91°C (decreased from 95°C)

2.1.2 Customer Affecting Changes

Contents:

2.1.2.1 Changes in This Release

This section provides a list of changes that took place in the current version and break compatibility/interface or discontinue support for features or OS versions, etc.

Change	Description	Customer Impact and Recommendation
Updated SEL format for thermal and voltage sensors	The SEL format for thermal and voltage sensors has been updated to use descriptive sensor names instead of numeric IDs. To ensure backward compatibility during this transition, the system now generates two SEL events for each occurrence: • Legacy event using the numeric sensor ID (e.g., `Temperature #0x24`) • New event using the descriptive sensor name (e.g., `Temperature thermal_p0`)	The legacy numeric ID format will be removed in a future release. Users are encouraged to update parsing tools and automation to utilize the new sensor name format.

2.1.2.2 Changes Planned for Future Releases

This section provides a list of changes that will take place in a future version of the product and will break compatibility/interface or discontinue support for features or OS versions, etc.

Planned for Version	Description
26.07	Feature "Secure Factory Reset" will add a nonce to the initiating command sent from the Host BMC. Users must be aware that: 1. Secure Factory Reset is considered BETA for 26.04 release. 2. Support of a nonce, in the VDM command 'Secure Factory Reset Request' will be required.

2.1.2.3 Changes in Earlier Releases

This section provides a list of changes that took place throughout the past two major releases that broke compatibility/interface or discontinued support for features or OS versions, etc.

Introduced in Version	Description	Customer Impact and Recommendation
N/A	N/A	N/A

2.1.2.4 Discontinued Features

List of features which are supported in previous generations of hardware devices:

- N/A

2.2 Supported Platforms and Interoperability

2.2.1 Supported NVIDIA BlueField-3 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

SKU	PSID	Description
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

2.2.1.1 Self-hosted BlueField-3 Platforms

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
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

2.2.2 Supported NVIDIA BlueField-2 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

NVIDIA SKU	Legacy OPN	PSID	Description
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; 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
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

NVIDIA SKU	Legacy OPN	PSID	Description
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

2.3 Bug Fixes in This Version

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

Reference	Issue Details
4995031503087 6	Description: Following a firmware update via the `Oobupdate` script, the version reported for components such as UEFI may occasionally fail to reflect the newly installed version. This is a transient reading error, not an update failure.
	Reported in version: 26.04
4662191	Description: The BMC pending version may be displayed in a non-standard format (e.g., the version `BF-26.01-36` may appear as `02601036`).
	Reported in version: 26.01
4917779	Description: Initiating an Arm `GracefulReset` can cause the BMC's Redfish `UpdateService` to incorrectly report its state as `UnavailableOffline`.
	Reported in version: 25.10-LTSU2
4944048503600 9	Description: When upgrading or downgrading between the 25.10-LTSU2 and 26.04 releases, repeated BMC reboots may, in rare cases, cause the `profile-manager` service to fail due to a malformed JSON file. This failure triggers a core dump of the service, populates core dump logs, and causes the `golden-image` service to become unresponsive.
	Reported in version: 25.10-LTSU2
4917779	Description: Initiating an Arm `GracefulReset` can cause the BMC's Redfish `UpdateService` to incorrectly report its state as `UnavailableOffline`.
	Reported in version: 26.01
4948318494555 4	Description: If a secondary BMC task (such as a log dump) is started after the BMC firmware update has been initiated, but before the installer's monitoring logic has attached to it, the installer may mistakenly track the secondary task. This tracking error causes the installer to misjudge the update's completion, which can cause the subsequent BMC reboot to fail and leave the new firmware in a pending, unactivated state.
	Reported in version: 26.01

Reference	Issue Details
4401488	Description: The BMC kernel enforces `CONFIG_STATIC_USERMODEHELPER`, which routes all usermode helper calls to `/sbin/usermode-helper`. Because this executable does not exist on the system, these calls fail and may cause undefined system behavior.
	Reported in version: 26.01
4905017	Description: When operating in NIC mode, a host power cycle may intermittently cause the UEFI to fail to retrieve BMC Redfish credentials. This results in a `DPU-BMC RF credentials not found` error and introduces a timeout delay during the boot sequence.
	Reported in version: 26.01
4969243	Description: When the `ENABLE_BMC_WAIT` flag is active on BlueField-3 DPUs, the BMC SEL may intermittently fail to record the complete UEFI boot progress following a host power cycle. This is a cosmetic issue and does not affect the functionality of the DPU.
	Reported in version: 26.01
4995032	Description: Redfish queries via `OobUpdate --show_all_version` intermittently returned empty strings for IPMB-backed properties (such as BOARD and BSP).
	Reported in version: 26.01
4867786	Description: During BFB installation, the Golden ARM image update may intermittently hang and fail via Redfish, logging a `golden_image_arm firmware update timed out` error.
	Reported in version: 26.01
4914053	Description: The BFB installer defaults to DHCP for the VLAN4040 interface. If no DHCP server is present, the request silently fails after a 300-second timeout, bypassing the static IP fallback and skipping all BMC-related firmware updates.
	Reported in version: 26.01
4924426	Description: Following a DPU reset, the `BaseMAC`, `BaseGUID`, and `Description` fields may incorrectly return as empty within the `redfish/v1/Systems/Bluefield/Oem/Nvidia` schema response.
	Reported in version: 26.01
4987307	Description: During BFB installations via Redfish, the task state may change to "Exception" before the specific error message is appended to the HTTP response payload. This results in incomplete error logs on the initial poll following a failure.
	Reported in version: 26.01
4980118	Description: The `set_emu_params.sh` script attempts to load the `mlxbf_ptm` (DPU Power Telemetry) driver, leading to continuous module load failures and log spam because `mlxbf_ptm` is a non-upstreamed debug driver that is unsupported on several operating systems.
	Reported in version: 26.01
4799519	Description: Accessing the `/redfish/v1/Managers/Bluefield_BMC` Redfish endpoint may time out, accompanied by repeated `ipmb-host` timeout errors in the console and significant delays (multiple minutes) when opening the UEFI "System Configuration" page.

Reference	Issue Details
	Reported in version: 26.01
4932328	Description: Excessive Common Platform Error Record (CPER) files in `/var/cper` can exhaust the BMC root partition space. This crashes the `entity-manager` service and causes IPMI commands like `ipmitool sdr` to fail.
	Reported in version: 26.01
4957197	Description: When external monitoring tools or scripts repeatedly query the BMC's Redfish interface using Basic authentication over extended periods, internal session resources fail to release properly. This memory leak eventually causes the BMC to lose network connectivity, even while the DPU management interface remains online.
	Reported in version: 26.01
4966472	Description: The BMC generates a warning log for `PLDM_Sensor_1_100` when the NIC temperature reaches the official 91°C upper non-critical threshold. This is an expected hardware alert for elevated temperatures, not a software defect.
	Reported in version: 26.01

2.4 Known Issues

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

Ref #	Issue Details
5015602	Description: When updating to DOCA 3.4.0, changes to the ERoT code require an ERoT reset command. This command may intermittently cause the DPU to hang and eventually reset halfway through the installation. This premature reset interrupts the update process, preventing the new BMC and ERoT code from activating, and aborts the Config Image and NIC firmware updates before completion.
	Workaround: Manually trigger a graceful restart of the ERoT via the following Redfish API, and then rerun the update process: `curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST <https://<bmc_ip>>/redfish/v1/Chassis/Bluefield_ERoT/Actions/Chassis.Reset -d '{"ResetType":"GracefulRestart"}'`
	Keyword: ERoT; DPU reset; firmware update
	Discovered in version: 26.04
4914053	Description: The BFB installer defaults to DHCP for the VLAN4040 interface. If no DHCP server is present, the request silently fails after a 300-second timeout, bypassing the static IP fallback and skipping all BMC-related firmware updates.

Ref #	Issue Details
	Workaround: Ensure a DHCP server is accessible during installation, or manually configure a static IP for the VLAN4040 interface to prevent the timeout. If using the updated installer script, static IP is now the default and DHCP must be explicitly enabled via `VLAN4040_DHCP_ENABLE` in the `bf.cfg` file.
	Keyword: DHCP timeout; static IP; BMC firmware
	Discovered in version: 26.04
4931912	Description: Executing `ipmitool sel clear` when the BMC System Event Log (SEL) is at 100% capacity successfully clears the log but fails to record the expected "Log area reset/cleared" event entry.
	Workaround: Clear the SEL before it reaches maximum capacity.
	Keyword: SEL; capacity limit
	Discovered in version: 26.04
4994932	Description: In rare cases, the PLDM firmware update process in NIC mode may fail during the activation phase due to a transient communication error on the device.
	Workaround: Reset the BMC and re-attempt the PLDM firmware update.
	Keyword: PLDM; firmware update; NIC mode
	Discovered in version: 26.04
4882705	Description: The `ENABLE_BMC_WAIT` configuration requires a power cycle of the BlueField DPU to take effect and function properly. If a user initiates a warm reboot of both the DPU Arm and the DPU BMC simultaneously, the system treats these as two separate, uncoordinated actions. As a result, the DPU Arm/UEFI does not wait for the BMC to become fully ready before proceeding with its own boot sequence.
	Workaround: Always perform a full cold boot of the host server or issue a complete DPU power cycle instead of relying on independent warm reboots of the subsystems.
	Keyword: BMC readiness; DPU Arm
	Reported in version: 26.01
4868713	Description: When a PLDM firmware update initiated from the Host BMC is canceled or fails, the CEC firmware version may remain stuck in a "pending" state.
	Workaround: Perform a full power cycle of the card (power off the host, execute an `ipmitool power cycle` from the DPU BMC, and then power the host back on). Alternatively, you can reset the CEC directly without a full power cycle by issuing a Redfish CEC self-reset command.
	Keyword: PLDM; CEC; firmware update
	Reported in version: 26.01
4870597	Description: When upgrading the BlueField-2 BMC from an older version that lacks Redfish Host Interface support (e.g., 2.8.2-x) to a newer version that includes this feature (version 23.10 or later), the Redfish communication channel between the BlueField BIOS and the BMC Redfish server fails to initialize automatically.
	Workaround: Perform a BMC factory reset immediately following the firmware upgrade.
	Keyword: Redfish host interface; factory reset
	Reported in version: 26.01

Ref #	Issue Details
4876483	Description: When upgrading the BlueField firmware bundle (e.g., from 3.1.0 to 3.3.0), the doca-installer post-installation verification may fail with return code 1. This is caused by a cosmetic mismatch in the BMC version string, where the pending version includes a leading zero (e.g., BF-26.01-04) while the expected target version does not (e.g., BF-26.01-4).
	Workaround: This error is harmless and can be safely ignored. A standard system power cycle resolves the mismatch.
	Keyword: BMC firmware; version mismatch
	Reported in version: 26.01
4885282	Description: When enabling Livefish mode from the DPU-BMC using `ipmitool`, the command may return a timeout error. This response is erroneous; the DPU successfully transitions into Livefish mode despite the error message.
	Workaround: N/A
	Keyword: Livefish
	Reported in version: 26.01
4857246	Description: The Redfish interface may list unsupported parameters (such as `EROT`, `ROT`, and `FirmwareAttributes`) under `OEMDiagnosticDataType`. If a user attempts to collect diagnostic data using these unsupported types, the task remains in a "Running" state for approximately 45 minutes before eventually failing or being canceled.
	Workaround: Use only the diagnostic dump types listed in this guide (typically `BMC` and `System`). Ignore unsupported values listed in the Redfish `CollectDiagnosticDataActionInfo` output.
	Keyword: CollectDiagnosticData; OEMDiagnosticDataType; task timeout; diagnostic dump
	Reported in version: 26.01
4502081	Description: When the BMC SEL is full or contains a large number of entries, the BMC may require more than 110 seconds to fully initialize the SDR, sensor, and FRU information after a reboot. In rare cases (1-2%), this delay may exceed expected timeouts, taking up to ~120 seconds.
	Workaround: N/A
	Keyword: Reboot latency; SEL
	Reported in version: 26.01
4850556	Description: Performing a power reset of the DPU via the DPU-BMC may result in partial IP address information being displayed in `redfish/v1/Systems/Bluefield/Ethernetinterfaces/`. This issue is cosmetic and does not impact network functionality.
	Workaround: Reboot the DPU-BMC to refresh and display the full IP address information.
	Keyword: Redfish; IP address
	Reported in version: 25.10
4877678	Description: Stress testing during upgrades and downgrades between BMC versions 25.01-1 and 24.10-17g may occasionally result in a loss of BMC IP.
	Workaround: N/A
424034	Keyword: Stress testing
	Reported in version: 25.10

Ref #	Issue Details
4127100	Description: The BMC System Event Log (SEL) capacity calculation incorrectly excludes critical and warning events, counting only informational (debug) entries. As a result, the total number of SEL entries may exceed the configured maximum limit (e.g., exceeding a 600-entry limit), leading to inaccurate capacity reporting in tools like `ipmitool sel info`.
	Workaround: Manually clear the SEL log if it becomes too large.
	Keyword: SEL; capacity limit
	Reported in version: 25.10
4502081	Description: When the BMC SEL is full or contains a large number of entries, the BMC may require more than 110 seconds to fully initialize the SDR, Sensor, and FRU information after a reboot. In rare cases (1-2%), this delay may exceed expected timeouts, taking up to ~112 seconds.
	Workaround: N/A
	Keyword: Reboot latency; SEL
	Reported in version: 25.10
4793060	Description: BlueField-3 fwbundle downgrade is not supported via PLDM in BMC 25.10-LTSU1.
	Workaround: N/A
	Keyword: Deferred update; downgrade
	Reported in version: 25.10
N/A	Description: The IP address range 192.168.240.0–192.168.240.7 is reserved for internal BMC–UEFI communication. User-configured TCP connections that utilize addresses within this range may be automatically terminated by the system.
	Workaround: N/A
	Keyword: TCP; connection
	Reported in version: 25.10
4422284	Description: When the Redfish eROT update command is used, the eROT image is updated regardless of the value set for the `ForceUpdate` flag.
	Workaround: N/A
	Keyword: eROT force update
	Reported in version: 25.10
4650613	Description: On BlueField-2 DPUs, the BMC may not switch to the updated firmware image after a host power cycle during upgrade. This occurs because the ERoT (CEC) resets during the cycle, preventing the BMC flash from activating the new image. The update becomes effective only after the BMC itself is rebooted.
	Workaround: Reboot the BMC after the host power cycle to activate the new image.
	Keyword: Firmware update; ERoT; CEC
	Reported in version: 25.10
4336172	Description: Pushing a PLDM package update to the BMC/CEC without rebooting the host causes the update process to fail.
	Workaround: N/A
	Keyword: PLDM; CEC

Ref #	Issue Details
	Reported in version: 25.07
4560376	Description: Performing a BMC factory reset without following up with a graceful reset leaves the BMC in an incomplete transitional state, resulting in undefined system behavior.
	Workaround: Ensure that after initiating a factory reset, a graceful reset is performed to complete the process. If skipped, re-run the factory reset procedure and immediately follow with a graceful reset.
	Keyword: Factory; graceful; reset
	Reported in version: 25.07
4029375	Description: The Redfish schema for `/Systems/Bluefield/ResetActionInfo` is not supported.
	Workaround: N/A
	Keyword: Redfish
	Reported in version: 25.04
4284756	Description: Boot option re-enabled by UEFI after power reset of DPU.
	Workaround: 1. Setting Boot option enable attribute to false. 2. Power reset DPU. 3. Do not reboot or power reset DPU for a second time. If reboot/power reset DPU does occur, the boot option enable attribute will be recovered to true on UEFI generated Boot Options, but this issue does not affect legacy boot options. 4. Completely disable boot options getting reset by UEFI by using `SYS_DISABLE_AUTO_BOOT_REFRESH=TRUE` configuration in bf.cfg.
	Discovered in version: 25.01
4259780	Description: Updating BFB image via HTTPS without CA on BlueField-2 will fail and may not return the correct failure message.
	Workaround: N/A
	Discovered in version: 25.01
4261446	Description: When the BlueField-2 SEL log is full, multiple attempts may be needed to load the SEL database and initialize the log.
	Workaround: Retry dump generation command.
	Discovered in version: 25.01
4276783	Description: Orchestrated DPU-BMC reboot will not work after upgrade if the CEC version has not changed.
	Workaround: N/A
	Discovered in version: 25.01
4177391	Description: BMC Redfish schema contains invalid attributes, causing the Redfish Mockup creator to return `elements not found` errors.
	Workaround: N/A
	Discovered in version: 25.01

Ref #	Issue Details
4243159	Description: BMC dump collection is missing `procfd.log`.
	Workaround: N/A
	Discovered in version: 25.01
4249132	Description: Copying a file to the BMC that consumes space in the `root` filesystem will render the BMC unusable after a reboot.
	Workaround: Do not copy any files to the BMC. Doing so will require the BlueField-3 to be swapped out.
	Discovered in version: 25.01
4178704	Description: When deleting an IPv4/6 static IP address via the Redfish interface, an unexpected failure message may occur even though the operation has actually succeeded. Example for an IPv6 address: `redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0 -d '{"IPv6StaticAddresses": [{"Address": "0000:0000:0000:0000::0000", "PrefixLength": 64}]}'`
	Workaround: N/A
	Discovered in version: 24.10-LTSU1
3922872	Description: In the `Systems/Bluefield/LogServices/Dump/Entries` Redfish schema, the `Originator` and `OriginatorType` fields are not persistent following a BMC reboot.
	Workaround: N/A
	Discovered in version: 24.10
4123370	Description: The error notification is not clear on BFB Update when RShim is not owned by the BMC.
	Workaround: N/A
	Discovered in version: 24.10
4048673	Description: The AllowableValues in the response for a GET request to the `Chassis/Bluefield_ERoT/ResetActionInfo` scheme are not updated by the BMC.
	Workaround: N/A
	Discovered in version: 24.10
4127461	Description: On BlueField-2, when the BMC comes up after reboot and the SEL buffer is full, it takes about 30 seconds for the BMC to finish setting up all of the SEL entries. Until then, adding more entries or querying the existing ones cannot be done.
	Workaround: N/A
	Discovered in version: 24.10
N/A	Description: Ownership of RShim may be ambiguous when ownership is forced by a new host causing several BMC features to not work due to the ownership conflict.
	Workaround: Reboot the BMC after moving RShim ownership to a new host. This ensures the BMC correctly synchronizes with the current RShim ownership status.
	Discovered in version: 24.07

Ref #	Issue Details
3877835	Description: `openbmctool` is no longer supported.
	Workaround: The supported management interfaces are IPMItool and Redfish.
	Discovered in version: 24.07
4027478	Description: BMC's IP aliasing feature allows multiple IP addresses to be assigned to a single network interface, which permits the interface to respond to different IP addresses.
	Workaround: N/A
	Discovered in version: 24.07
3995907	Description: The Redfish schema for BlueField data ports (`Systems/Bluefield/EthernetInterfaces`) displays only the valid IP and MAC addresses. Other attributes associated with this schema should be ignored and considered as not current.
	Workaround: N/A
	Discovered in version: 24.07
3995907	Description: The update to `ipmitool` version 1.8.19 has introduced inconsistencies between the tool and the BMC implementation, leading to incomplete information being displayed when executing the `lan print` command.
	Workaround: N/A
	Discovered in version: 24.07
3995899	Description: The DPU BMC is equipped to detect ECC errors from the BlueField Arm system and record these errors in the BMC SEL. For SEL events related to single-bit ECC errors, the BMC does not recognize the resolution status on the BlueField Arm and consequently logs `"Resolved": false` irrespective of the actual state.
	Workaround: N/A
	Discovered in version: 24.07
3925815	Description: When changing the BMC IPv6 address configuration from static to DHCP, the static IP address in the IPv6 list does not get disabled.
	Workaround: N/A
	Discovered in version: 24.07
3899719	Description: After executing `sel clear` , the event log in `LogServices` generates `Error.LogsCleared` .
	Workaround: N/A
	Discovered in version: 24.07
4005373	Description: DPU BMC does not support the following attributes and schemas within the `Systems/Bluefield` Redfish schema: `MemorySummary` , `Memory` , `Processors` , `Storage` , `GraphicalConsole` , and `PowerMode` .
	Workaround: N/A
	Discovered in version: 24.07

Ref #	Issue Details
3995906	Description: The BlueField power cycle relies on the PCIe PERST/ALL_STANDBY. If the host does not disconnect the PCIe link, the DPU BMC would cancel the power cycle request to avoid causing server errors. Since the power cycle command is asynchronous, it always returns success; however, if it fails, a new SEL log is created to indicate that the command has not executed.
	Workaround: Get the status from the BMC SEL.
	Discovered in version: 24.07
3998661	Description: The `ipmitool set static ip` command deletes both IPv4 and IPv6 then creates IPv4 static IP, then changes the IPv4 address causing 1 more create and delete events.
	Workaround: N/A
	Discovered in version: 24.07
3914629	Description: When "Force PXE" is set right before installing a BFB image via BMC RShim, BlueField PXE boot fails to boot from NET-OOB-IPV4.
	Workaround: N/A
	Discovered in version: 24.04
4064373	Description: The DPU BMC LLDP represents the eth0 interface. If the user initiates a VLAN interface on top of eth0, the LLDP schema does not function as expected and the transmitted data does not accurately describe the eth0 attribute or the newly created VLAN.
	Workaround: N/A
	Discovered in version: 24.04
3994990	Description: During a power cycle of the BlueField, the `ipmb_host` driver running on the BlueField OS may crash due to communication issues with the BlueField BMC, causing the BlueField OS to become stuck.
	Workaround: Perform an additional reboot or reset to the BlueField.
	Discovered in version: 24.01
3725502	Description: Recording user operations takes place regardless of the action's success.
	Workaround: N/A
	Discovered in version: 24.01
3598450	Description: The boot process may stall following a non-graceful reboot.
	Workaround: Do not issue force reboot to the BlueField BMC.
	Discovered in version: 24.01
3664596	Description: When declaring only a secondary NTP server, this server acts as the primary one.
	Workaround: N/A
	Discovered in version: 24.01
3747285	Description: The ipmitool command to force PXE in BMC modifies both the IPMI and Redfish request settings. When Redfish is enabled in UEFI, Redfish takes priority, so all PXE boot entries are attempted and before regular boot continues.
	Workaround: Redfish must be disabled if IPMI force PXE retry behavior is expected.
	Discovered in version: 24.01

Ref #	Issue Details
3662417	Description: The Arm UEFI is crucial for rapid system booting, activating the NIC for host network communication. The Arm UEFI tries to fetch Redfish host credentials from the DPU BMC during boot-up. If not done within a designated timeframe, UEFI skips Redfish setup and boots the OS, ignoring any Redfish server settings or updates.
	Workaround: Reboot the BlueField Arm core again.
	Discovered in version: 24.01
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
35341150	Description: The BMC and BlueField 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
36311999	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
36549300	Description: If the BlueField BMC firmware has been upgraded from version 2.8.2-x or older to version 23.03 or newer, it is necessary to execute a factory reset of the BlueField BMC.
	Workaround: N/A
	Discovered in version: 23.03
36375527	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
36346649	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

Ref #	Issue Details
3634603	Description: When the BlueField 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
3609525	Description: Following a reboot of BlueField's BMC, it is necessary to wait 30 seconds to allow for the complete loading of system services before initiating a reboot of BlueField itself.
	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 BlueField. 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 BlueField and BMC boot independently which may lead to BlueField'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

2.5 Bug Fixes History

Ref #	Issue Details
4826162	Description: On systems equipped with multiple BlueField-3 DPUs, the `doca-installer --compare` command may intermittently fail to retrieve the CEC firmware version from one of the cards.
	Fixed in version: 26.01
4662191	Description: The BMC pending version may be displayed in a non-standard format (e.g., the version BF-24.10-36 may appear as 02410036).
	Fixed in version: 25.10-LTSU1
4658475	Description: The BMC pending version string generated by the BMC may include an unwanted leading zero in the last segment of the version (e.g., 00250804 is exposed as BF-25.08-04 instead of BF-25.08-4).
	Fixed in version: 25.10-LTSU1
4686938	Description: After a successful BFB update via SCP, subsequent updates may show incorrect progress messages (e.g., `Transfer progress: 100%`) that do not reflect the actual update status.
	Fixed in version: 25.10-LTSU1
4664471	Description: When creating a VLAN using `ipmitool`, the interface incorrectly appeared as a second entry in the `DedicatedNetworkPorts` collection. This results in redundant port entries showing identical LLDP data.
	Fixed in version: 25.10-LTSU1
4064373	Description: The DPU BMC LLDP represents the `eth0` interface. If a VLAN interface is initiated on top of eth0, the LLDP schema does not function as expected and the transmitted data does not accurately describe the eth0 attribute or the newly created VLAN.
	Fixed in version: 25.10
4365835	Description: There is a mistake in the message displayed when the BISO secure boot setting is changed through the DPU BMC where the event log shows `ScureCurrentBoot` instead of `SecureBootCurrentBoot`.
	Fixed in version: 25.10
4549368	Description: Power sensors available via `ipmitool sensor` command are not available when DPU is in kernel lockdown mode.
	Fixed in version: 25.10
4551450	Description: After BMC factory reset, the ipmitool network interface may fail to connect to the DPU BMC via the RMCP interface.
	Fixed in version: 25.10
4525138	Description: SoC power sensors are not reported via BMC, resulting in missing power telemetry data in ipmitool and WebUI.
	Fixed in version: 25.07
4484568	Description: BMC fails to boot due to `No Space Left` on `/var/lib/phosphor-debug-collector/dumps` directory, leading to inaccessible BMC.
	Fixed in version: 25.07

Ref #	Issue Details
4458182	Description: If a corrupted or unauthenticated BFB image is transferred from the BMC to the DPU, the system halts the installation process as part of a built-in security mechanism. Once triggered, the recovery path remains locked to prevent potential compromise.
	Fixed in version: 25.07
4484640	Description: Certain BMC users fail to authenticate via IPMI lanplus interface (RAKP unauthorized name error).
	Fixed in version: 25.07
4517562	Description: After upgrading the BMC to version 24.10 or later, SEL entries generated by earlier versions may no longer appear in ipmitool sel output.
	Fixed in version: 25.07
4530693	Description: Over-temperature warnings in ipmitool SEL logs lack port identifiers and show inconsistent sensor IDs compared to WebUI/Redfish, complicating troubleshooting.
	Fixed in version: 25.07
4331648	Description: If user sets SEL capacity to a lower size than the previous setting, some existing SEL entries might be erased rather than preserved.
	Fixed in version: 25.04
4376078	Description: IPMI SEL is missing output while WEBUI SEL displays in full.
	Fixed in version: 25.04
4052874	Description: `UefiSignatureOwner` field is not supported. If this field is populated with data, an exception occurs.
	Fixed in version: 25.01
4158416	Description: When updating the BFB image on a BlueField-2 system via HTTP/HTTPS using Redfish, the operation may fail if the system is overloaded.
	Fixed in version: 25.01
4128189	Description: When updating the BFB image on a BlueField-2 system via HTTP/HTTPS using Redfish, the operation may fail if the system is overloaded.
	Fixed in version: 24.10-LTSU1
4129718	Description: If the path of installation of the bfb image is called and the RShim on the host is not connected, the BMC takes the RShim. If the RShim on the host is connected, calling this path returns an error.
	Fixed in version: 24.10-LTSU1
4135001	Description: The serial number is missing from SMBIOS table 3 and from Redfish schema `/Chassis/Card1`.
	Fixed in version: 24.10-LTSU1
4151178	Description: The `ipmid` and `netipmid` processes have an inconsistent view of the user database until changes propagate to `netipmid`.
	Fixed in version: 24.10-LTSU1
4047689	Description: While running the reprovisioning script from BMC, the RShim boot device appears to be busy which causes the script to fail without completing the process.
	Fixed in version: 24.10

Ref #	Issue Details
4146640	Description: In the event of a server reboot, the BMC may boot before the host and take control of the RShim before the host.
	Fixed in version: 24.10
3991930	Description: The reported dump entry creation date is not initialized properly and reports the default system date `1970-01-01T00:28:43.991149+00:00` when creating a dump entry using the LogService on the Redfish interface.
	Fixed in version: 24.10
4064371	Description: The BMC dump collection is missing the `varfilelist.log` and `slabinfo.log` files.
	Fixed in version: 24.10
3906500	Description: Using the header `connection: close` in a Redfish request terminates the `x-auth-token` session
	Fixed in version: 24.07
3875280	Description: UUID and SKU properties are intermittently unavailable after BlueField BMC reboot.
	Fixed in version: 24.07
3888140	Description: When warm rebooting the BlueField OS, the IPMB channel between the BlueField and BlueField BMC may fail to function due to underlying I2C channel issues. If this occurs, all functionality relying on this channel are affected, including: • IPMI commands from the BlueField OS to the BlueField BMC and vice versa • Redfish BlueField inventory schema • Redfish network schema of the BlueField OOB and network interfaces • BlueField sensor information
	Fixed in version: 24.07
3878990	Description: The BMC may provide incorrect bootstrap credentials to the UEFI. This results in the failure of any BIOS configurations.
	Fixed in version: 24.07
3855648	Description: The manager factory reset action is named `ResetToDefaults` instead of `ResetType` as per the Redfish Data Model specification.
	Fixed in version: 24.07
3599016	Description: After a BFB update, it takes the BMC ~30 seconds to sync with the true values from the DPU reflected in the command `ipmitool sensor list`.
	Fixed in version: 24.04
3837485	Description: In some instances, consecutive core dumps occur, and since extracting the NIC debug log is a lengthy operation, this could result in log mismatch and inaccurate information.
	Fixed in version: 24.04
3780188	Description: Add bad syndrome pipe also in dynamic mode.
	Fixed in version: 24.04
3634701	Description: In the Redfish `Systems/Bluefield` schema, the `Description` attribute is of a generic type and does not specify the DPU system.
	Fixed in version: 24.04

Ref #	Issue Details
3662417	Description: The BMC may provide incorrect bootstrap credentials to the UEFI. This would result in the failure of any BIOS configurations.
	Fixed in version: 24.01
3715528	Description: The `TransferProtocol@Redfish.AllowableValues` under the `simpleUpdate` service is held in a double list, deviating from the DMTF definition.
	Fixed in version: 24.01
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
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

Ref #	Issue Details
3554128	Description: `dmidecode` output does not match `ipmitool fru print` output.
	Fixed in version: 23.07
2930671	Description: A power cycle of the system might result in BMC MAC change.
	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

2.6 Change Log History

2.6.1 Changes and New Features in 26.04

- Update bluefield_temp sensor thresholds from 105C Critical High to 96C Critical High, and from 95C Warning High to 91C Warning High.

2.6.2 Changes and New Features in 26.01

- The SEL format for thermal and voltage sensors has been updated to use descriptive sensor names instead of numeric IDs. Refer to "Customer Affecting Changes" for information on backward compatibility.
- Added Redfish OOB support for configuring NIC Host Restriction settings, enabling administrators to enforce Zero Trust policies (equivalent to `mlxprivhost`) without logging into the DPU OS.

2.6.3 Changes and New Features in 25.10-LTSU1

- Enhanced Redfish connection management between BMC and UEFI to ensure proper termination of all open TCP connections
- Added support for querying ATF, UEFI, and firmware versions from BMC inventory independent of DPU OS availability
- Updated temperature AEN sensor SEL logging to use descriptive sensor names instead of generic sensor IDs
- Improved BMC log dump performance to reduce execution time:
    - Changed the default BMC dump file format from `.tar.xz` to `.zstd`. This update improves dump generation performance.

2.6.4 Changes and New Features in 25.10

- Implemented a deferred firmware update flow that enables updating all firmware components from the DPU-BMC without service interruption in DPU mode
- Enhanced security through mandatory kernel module signing and verification, ensuring all kernel modules are cryptographically signed and verified by the Linux kernel before loading
- Added user-accessible functionality to force BMC RShim ownership, enabling recovery scenarios where manual intervention is required to reclaim RShim control for troubleshooting and system recovery operations

2.6.5 Changes and New Features in 25.07

- Added Redfish system/processor schema support for BlueField-3 to provide comprehensive hardware inventory information through standard Redfish interfaces (System Processor)
- Implemented ATX power loss SEL event reporting to BMC during AC power cycles for BlueField-3 systems to improve power state monitoring and diagnostics (ATX Power Error)
- Enhanced ipmitool bootdev commands with automatic 60-second invalidation when no reboot or power reset execution occurs on host (BlueField Arm) (Boot Source Override Config Using IPMI)
- Support live firmware upgrades, enabling firmware updates without hots (BlueField Arm) downtime (section "BFB Live Firmware Update" in Installation for DPU Mode)

2.6.6 Changes and New Features in 25.04

- Extended Arm-UEFI support to ensure seamless operation until BlueField BMC initialization completes
- Added RTC battery voltage monitoring to the SDR list for enhanced system diagnostics
- Implemented Redfish mutual authentication support for BlueField-3 platforms
- Updated BMC FRU content to enhance backward compatibility
- Introduced a unified, time-synchronized logging system for BlueField-3 BMC to ensure consistent event order and accurate timestamps across IPMI and Redfish interfaces. The system will begin assigning new SEL IDs starting from 1. If existing SEL entries have overlapping IDs, they will be overwritten by new entries. After upgrading to the new version, it is advised to clear all SEL entries to prevent potential ID conflicts and ensure optimal log integrity.
- Added BMC Redfish support for remote attestation over Redfish specifically for SPDM:
  - BlueField NIC
  - CEC1736 (BMC ERoT)
- Added support for the sensors `ddr_temp` and `rtc_voltage` under "BMC Sensor Data"
- Security Hardening: Implemented several Linux kernel configuration changes to improve system security and activated kernel module signature. The following table summarizes key modifications:

Parameter	Old Value	New Value	Reason
`CONFIG_KEXEC`	`yes`	Not set	Enables replacement of running kernel using `kexec` command.

Parameter	Old Value	New Value	Reason
`CONFIG_SLAB_MERGE_DEFAULT`	`yes`	Not set	Prevents merging similar-sized slab caches, mitigating cross-slab heap attacks
`CONFIG_SHUFFLE_PAGE_ALLOCATOR`	Not set	`yes`	Enables randomization of the high-order page allocation freelist
`CONFIG_SECURITY_DMESG_RESTRICT`	Not set	`yes`	Prevents kernel memory address leakage through `dmesg`
`CONFIG_DEBUG_FS`	`yes`	Not set	Disables `debugfs`, reducing the kernel's attack surface
`CONFIG_BPF_SYSCALL`	`yes`	Not set	Disables the `bpf()` syscall, restricting manipulation of BPF programs and maps
`CONFIG_USER_NS`	`yes`	Not set	Disables user namespaces to prevent privilege escalation via namespace exploits
`CONFIG_BUG_ON_DATA_CORRUPTION`	Not set	`yes`	Enables kernel validation checks for detecting data corruption
`CONFIG_DEFAULT_MMAP_MIN_ADDR`	`4096`	`32768`	Increases the minimum mmap address to mitigate kernel NULL pointer dereference exploits
`CONFIG_DEBUG_KMEMLEAK`	`yes`	Not set	Disabled due to its dependency on `CONFIG_DEBUG_FS`, which is also now disabled This parameter changed only in BlueField-2 (already not set in BlueField-3).

Changes to the kernel configuration parameters were made in accordance with recommended security hardening practices from the Linux Kernel Self-Protection Project (KSPP), grsecurity and CLIP OS.

2.6.7 Changes and New Features in 24.10 LTSU2

- System now logs comprehensive power events (CPER and SEL) in the DPU BMC if ATX power is disconnected from the BlueField-3 card during card operation.
- Added a dedicated sensor to the DPU BMC sensor suite that provides real-time visibility into SoC power consumption.
- Added support for capturing BlueField-3 PCIe errors and presenting them in the DPU BMC's CPER and SEL logs, improving troubleshooting and diagnostics.

2.6.8 Changes and New Features in 25.01

- Bug fixes

2.6.9 Changes and New Features in 24.10 LTSU1

- Added Redfish support to get OOB network 3-port switch port link status

- Added current limitation alarm events to BMC system event log (SEL)
- Integrated UE/CE memory/CPU cache ECC error recording into BMC CPER logger
- Added support for ipmitool commands to configure IPv6 gateway
- Added support for RAS logging

2.6.10 Changes and New Features in v24.10

- To ensure FRU device information is consistent across different system components, NVIDIA Networking is aligning the BlueField BMC FRU board product name with the system product name. This means that there would be two variations of BlueField BMC FRUs in the field. The following are the supported deviations:

FRU Field	Rev-1 (Old)	Rev-2 (New)
FRU Device Description	Nvidia-BMCMezz (ID 169)	BlueField-3 DPU (ID 243)
Board Manufacturing Date	<Board-mfg-date>	<Board-mfg-date>
Board Manufacturer	Nvidia	Nvidia
Board Product	Nvidia-BMCMezz	BlueField-3 DPU
Board Serial	<Board-serial>	<Board-serial>
Board Part Number	<Board-part-number>	<Board-part-number>

- Security enhancements – Introduced a new BIOS option to disable the IPMI channel between the BlueField Arm cores and its BMC. By default, this interface is enabled. However, a new BIOS attribute accessible via Redfish now permits disabling this interface for enhanced security.
- IPMB channel relocation – The IPMB channel used by the BlueField BMC to retrieve data from the BlueField Arm is now utilizing a dedicated I2C interface. This change is aimed at improving the serviceability of the interface.
- Temperature monitoring – Added the BlueField Arm DDR sensor to the BlueField BMC sensor list. This addition enables the monitoring of DDR temperature (ddr_temp), ensuring better performance and reliability.
- Event logging enhancements – Introduced several new event alarms:
  - BlueField data port module over current event (Module Critical Power Consumption)
  - BlueField data port module temperature event (Module Temperature Going High/Low)
  - BlueField Arm frequency change event (Arm Frequency Change)

    These alarms are now logged to the BMC System Event Log (SEL) for better event tracking and system diagnostics.
- BIOS Debug Mode – An option has been added to enable BIOS debug mode, facilitating advanced troubleshooting and system analysis

2.6.11 Changes and New Features in v24.07

- Extended DHCP mode setting to provide control for each IP version. In the current version, IPMITool includes a dedicated function to control the mode for both IPv4 and IPv6. For more details, please refer to section "Configuring IPv6 Mode".
- Updated Linux kernel from version 5.15 to 6.1
- Upgraded BlueField BMC Linux packages:
  - libpam 1.6 to 1.61
  - curl 8.5 to 8.7.1
  - bash 3.2.57 to 5.2.21
  - DNSmasq 2.9
  - glibc 2.39
  - ipmitool 1.8.19
  - busybox 1.36.1
  - rsyslog 8.2402.0
- The DPU BMC no longer supports openbmctool; all APIs are now accessible via Redfish
- The Redfish schema at `/redfish/v1/Cables/` is no longer supported. The data port link state is now accessible through the `Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/Ports` schema which includes the link state for the available data ports.
- The size of the BMC dump entry container is no longer limited to only two BMC dump entries. The limit now applies to the total amount of memory stored in the container, thus allowing more BMC dump entries to be stored in it depending on their size (Creating BMC Dump Task).
- Added support for the IPMI OEM command to configure the guest tunnel (Guest Tunnel)
- Extended the BMC log to capture Redfish/IPMI command configurations initiated by the user of the BMC (System Logs)
- RAS record UE/CE faults of MEM into BMC SEL (RAS Errors)
- Enhanced Redfish BFB SimpleUpdate to support HTTP/HTTPS file transfer policy (Installing BFB)
- Introduced rsyslog capability to log BMC SEL entries and Arm console output to a remote server (Rsyslog)
- Network re-provisioning (Bare-metal Reprovisioning):
  - Added an option in network re-provisioning for BMC to halt instead of reset after provisioning, allowing users to choose when to reset the DPU after provisioning is complete
  - Added support for ATF/UEFI and NIC firmware golden image versioning
- Added support for RShim force ownership request

2.6.12 Changes and New Features in v24.04

- Updated RShim user space driver to version 2.0.27
- Added an additional FRU device to the DPU BMC to reflect DPU Arm FRU information (System FRU)
- Introduced a redfish command to reset the DPU BMC eROT (Glacier) (Activating New CEC)
- Implemented support for sending sysrq controls to the DPU Arm through the SOL interface (SysRq Support in SOL)

- Change the Redfish BFB software update and Redfish system dump to try automatically to acquire the RShim interface
- Added Redfish schema to provide DPU description, base MAC, and base GUID (DPU Information)
- Introduced Redfish schema to support `MultipartHttp UpdateService` for DPU BMC and CEC Firmware update (BMC and CEC Firmware Operations)
- Added Redfish schema for the BlueField Arm network interfaces (OOB and data ports) (BlueField Host Network Interface)
- Included LLDP Redfish schema for the BlueField BMC's 1Gb/s interface (LLDP in Redfish)
- Enhanced Redfish support for the service identification property (Product Instance Identifier)
- Enforced ipmitool user privilege policy

2.6.13 Changes and New Features in v24.01

- Expanded the BMC dump log to incorporate data from the NIC firmware. A new log containing NIC device debug information has been introduced and is now accessible on NVIDIA® BlueField®-3 systems.
- Enabled DPU BMC to facilitate soft shutdown requests to the embedded CPU via both IPMI and Redfish protocols
- Upgraded the Linux Kernel version to 5.15 in the OpenBMC system
- Added IPMI and Redfish commands to disable/enable DPU Arm out-of-band (OOB) access to the management network
- Added new entries to the BMC system event log (SEL) or BMC operation log, enhancing support for BMC operations
- Incorporated a Redfish command for the deployment of BIOS CA certificates
- Updated BMC password policy
- Added support for simple `HttpMultiPart` update for BMC and eROT firmware

2.6.14 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 – the 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.

  ⚠ 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

2.6.15 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

2.6.16 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

3 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 NVIDIA® BlueField® using Redfish commands. However, IPMI commands and sysfs monitoring infrastructure are available as well.

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 BlueField'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

3.1 BlueField DPU

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

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

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

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

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

3.2 BlueField SuperNIC

The NVIDIA® BlueField® SuperNIC is an advanced network accelerator specifically engineered to optimize hyperscale generative AI workloads. Featured on the NVIDIA® Spectrum-X™ networking platform, the BlueField-3 SuperNIC is integrated across NVIDIA's accelerated systems to deliver peak AI workload efficiency and secure cloud multi-tenancy.

Powered by the NVIDIA DOCA software framework, the SuperNIC provides deterministic, isolated performance tailored for demanding cloud environments. It delivers up to 400 Gb/s of high-speed connectivity between GPU servers, utilizing advanced network acceleration features such as RoCE adaptive routing, out-of-order packet handling, and programmable congestion control. Furthermore, its half-height, half-length (HHHL) form factor and low-power profile ensure universal compatibility and seamless physical integration into most enterprise-class servers.

To read more about the BlueField-3 features and benefits, refer to this page.

To read more about the BlueField-2 features and benefits, refer to this page.

4 Connecting to BMC Interfaces

4.1 BMC Management Interface

The BMC has a separate Ethernet interface which provides network connection for management traffic to the BMC. The NVIDIA® BlueField® networking platform's bracket (DPU or SuperNIC) 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 BlueField's board label. By default, the IP address used for `eth0` is acquired via DHCP but can be configured differently.

4.1.1 BMC Password Policy

The BMC password must comply with the following policy parameters:

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

> Two characters are consecutive if `|hex(char_1)-hex(char_2)|=1`.
>
> Examples of passwords with 5 consecutive character pairs (invalid):
> `DcB a123456AbCd!` ; `ab1XbcYcdZdeGef!` ; `Testing_123abcgh!`.

The following is a valid example password:

- `HelloNvidia3D!`

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

4.1.2 Changing Default Password

When initially logging into the system, it is mandatory to update the default BMC password, `0penBmc`. The BlueField 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://<bmc_ip>/redfish/v1/
AccountService/Accounts/root -d '{"Password" : "<password>"}'
```

4.1.3 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}'
```


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
```

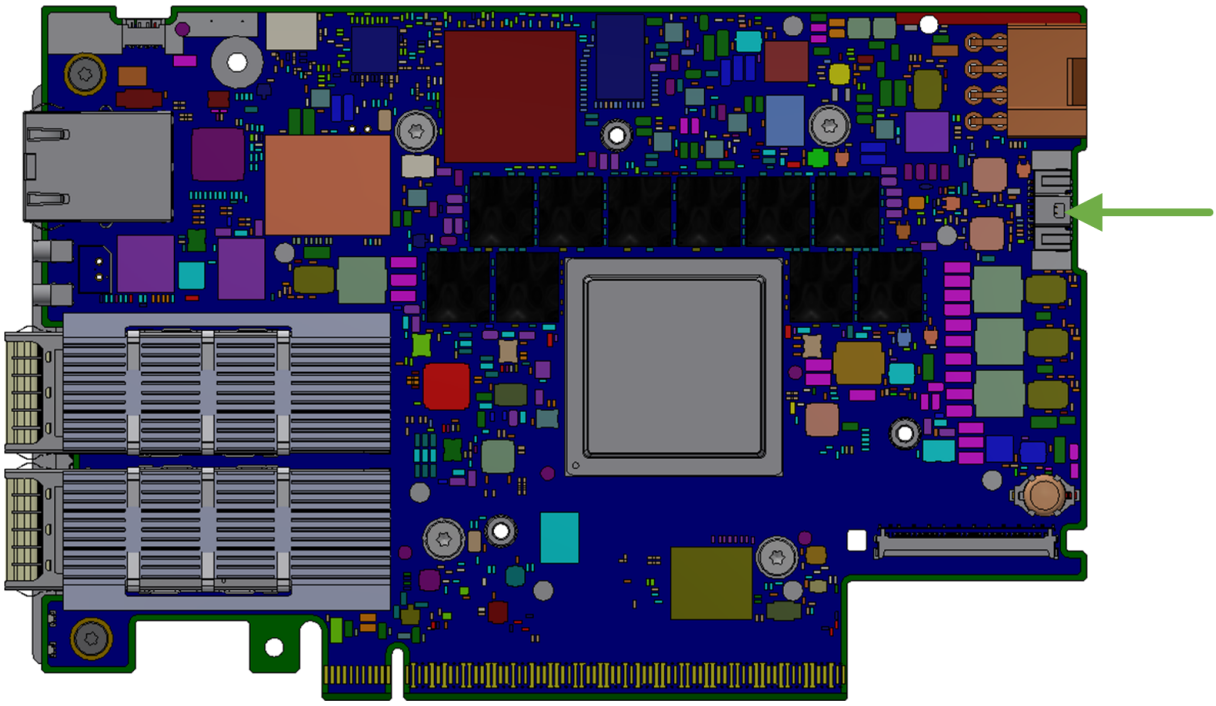
Example output:

```
{
  "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'."
  }
}
```

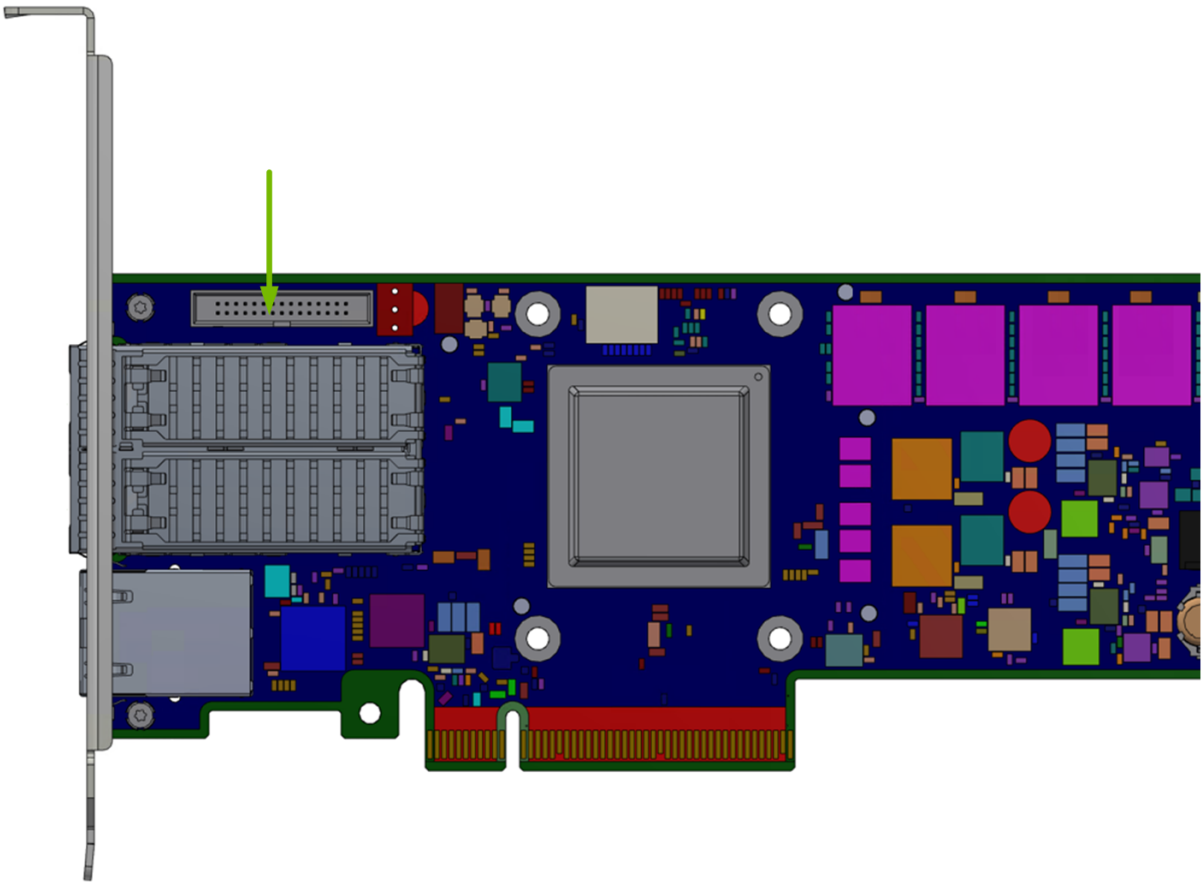
4.2 BMC Console Interface

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



4.3 Network Configuration

> 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.

4.4 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.

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

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

4.4.1 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.

5 Platform Management Interface

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

5.1 Redfish Management Interface

The BlueField'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
```

5.1.1 Redfish Implementation for NVIDIA BlueField BMC

Redfish Specification DSP0266 defines the necessary protocols, data model, behaviors, and other architectural components for creating an interoperable, multivendor, remote, and out-of-band capable interface. This interface is designed to meet the scalable platform management expectations of cloud-based and web-based IT professionals.

The specification outlines the mandatory elements required for all Redfish implementations and the optional elements that system vendors and manufacturers can choose to include. It also specifies where implementations can provide OEM-specific extensions.

NVIDIA BlueField BMC's Redfish implementation is based on the bmcweb implementation from the OpenBmc open-source community, ensuring compliance with the Redfish specification. As part of the BlueField development cycle, the code is synchronized with the upstream community, making it subject to changes and modifications derived from the upstream implementation. While NVIDIA is committed to adhering to the Redfish specification, the actual implementation may vary due to this development process.

5.1.2 Redfish POST (Action)

The bmcweb POST operation implementation complies with the DSP0266 specification definition. For more information, refer to the "POST (action)" section of Redfish Specification DSP0266.

5.2 IPMI

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

5.2.1 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>
```

5.2.2 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:

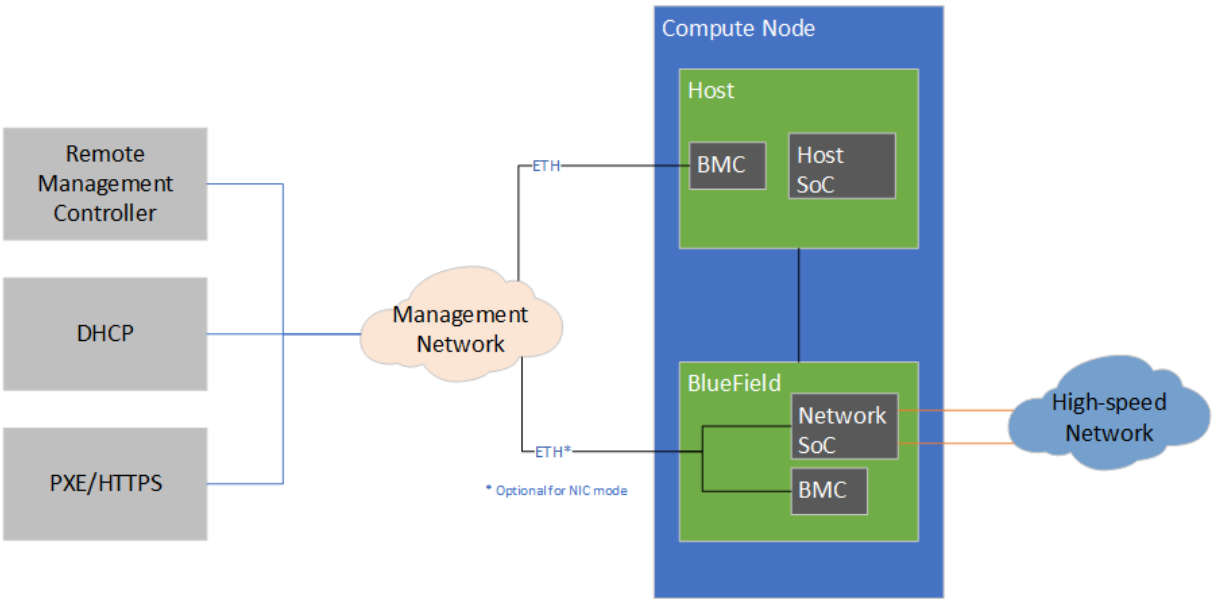
```
ipmitool -C 17 -I lanplus -H <bmc_ip> -U ADMIN -P ADMIN <ipmitool_arguments>
```

6 First-time Installation Procedure

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

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

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



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

- DPU mode installation (default mode for BlueField DPUs)
- NIC mode installation (default mode for BlueField SuperNICs)

6.1 DPU Mode Installation

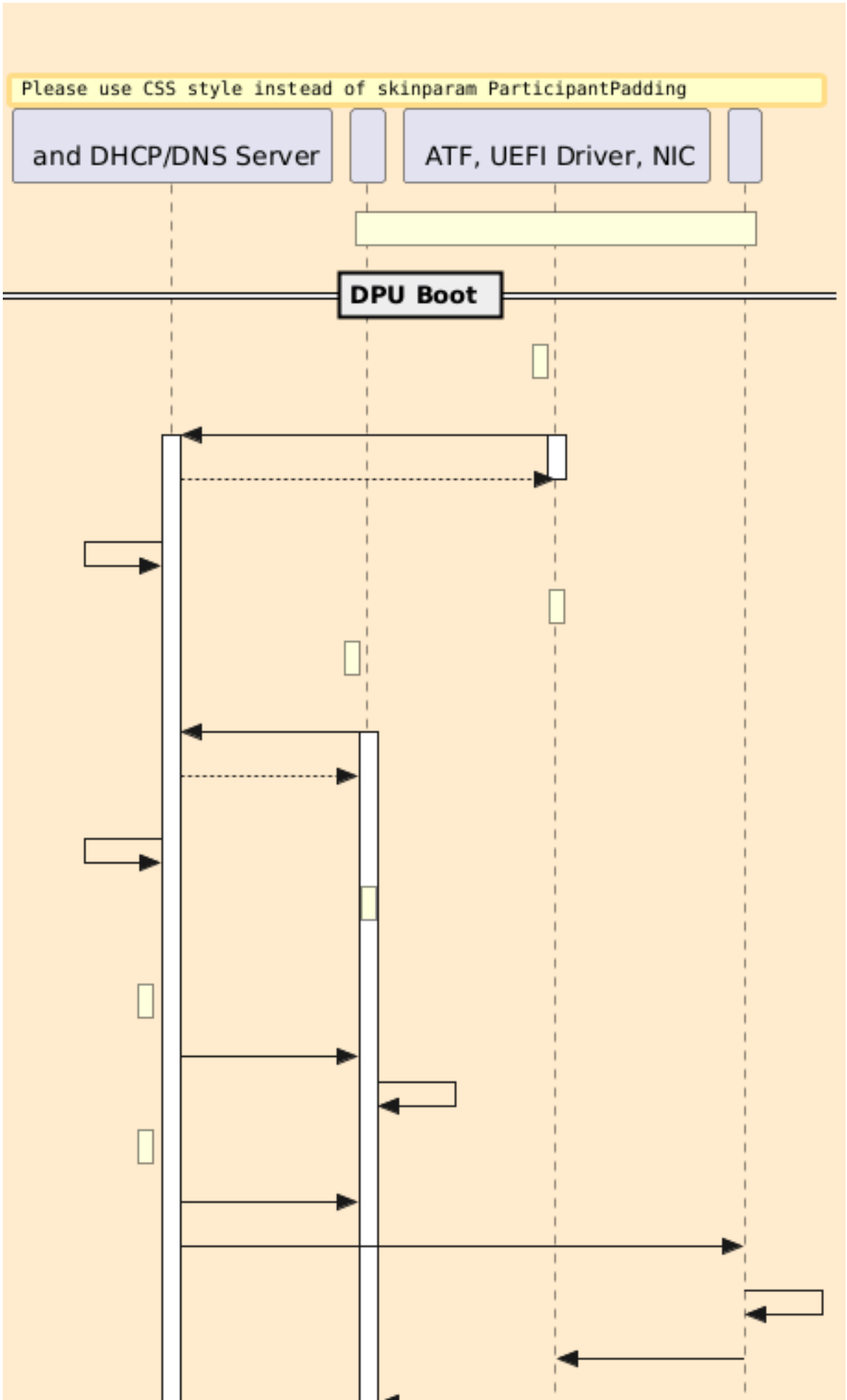
DPU mode is the default mode for BlueField DPUs, while BlueField SuperNICs are shipped with NIC mode as their default. To switch between the modes, see NVIDIA BlueField Modes of Operation. To check which mode your BlueField is currently running, see Common Configurations.

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

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

If a BlueField-2 is in your possession and it is the first time you are upgrading BlueField BMC, follow the instructions in appendix "BMC and eROT Upgrade Process for BlueField-2".

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


Please use CSS style instead of skinparam ParticipantPadding
and DHCP/DNS Server
ATF, UEFI Driver, NIC
DPU Boot

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

6.1.1 Step 1 – BlueField SoC Boots

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

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

6.1.2 Step 2 – BlueField BMC Boots

BlueField BMC issues DHCP DISCOVER over the 1GbE OOB interface, including vendor class (`"NVIDIA/BF/BMC"`) for BlueField-BMC, and MAC for identification and discovery. Example of BlueField BMC DHCP DISCOVER packet structure (note `"NVIDIA/BF/BMC"` in line 13):

```
1  root@bf-bmc:~# 18:18:10.563269 IP (tos 0xc0, ttl 64, id 0, offset 0, flags [none], proto UDP (17),
   length 320)
2  0.0.0.0.bootpc > 255.255.255.255.bootps: [udp sum ok] BOOTP/DHCP, Request from b8:3f:d2:ca:4b:26 (oui
   Unknown), length 292, xid 0xfc2acdec, secs 1, Flags [none] (0x0000)
3  Client-Ethernet-Address b8:3f:d2:ab:cd:ef (oui Unknown)
4  Vendor-rfc1048 Extensions
5  Magic Cookie 0x63825363
6  DHCP-Message (53), length 1: Discover
7  Client-ID (61), length 7: ether b8:3f:d2:ab:cd:ef
8  Parameter-Request (55), length 9:
9  Subnet-Mask (1), Default-Gateway (3), Domain-Name-Server (6), Hostname (12)
10 Domain-Name (15), Static-Route (33), NTP (42), Unknown (120)
11 Classless-Static-Route (121)
12 MSZ (57), length 2: 576
13 Hostname (12), length 7: "bf-bmc" Vendor-Class (60), length 13: "NVIDIA/BF/BMC" END (255), length 0
14 18:18:10.565261 IP (tos 0x0, ttl 63, id 0, offset 0, flags [DF], proto UDP (17), length 353)
15 (example) dhcp01.XX.YY > ldev-platform-13-043-bmc.bootpc: [no cksum] BOOTP/DHCP, Reply, length 325,
   hops 1, xid 0xfc2acdec, secs 1, Flags [none] (0x0000)
16 (example) Your-IP ldev-platform-13-043-bmc.XX.YY
17 (example) Server-IP l-pxe02.XX.YY
18 Gateway-IP 10.237.0.255
19 Client-Ethernet-Address b8:3f:d2:ab:cd:ef (oui Unknown)
20 file "pxelinux.0" Vendor-rfc1048 Extensions
21 Magic Cookie 0x63825363
22 DHCP-Message (53), length 1: Offer
23 Server-ID (54), length 4: (example) dhcp01.XX.YY
24 Lease-Time (51), length 4: 43200
25 Subnet-Mask (1), length 4: 255.255.0.0
26 Default-Gateway (3), length 4
27 (example) GW.XX.YY
28 Hostname (12), length 24: "ldev-platform-13-043-bmc" Domain-Name (15), length 13: "<local domain name>"
    NTP (42), length 4: (example) NTP.XX.YY
29 END (255), length 0
30 18:18:10.565261 IP (tos 0x0, ttl 62, id 0, offset 0, flags [DF], proto UDP (17), length 353)
31 dhcp01.XX.YY > ldev-platform-13-043-bmc.<local domain name>: [no cksum] BOOTP/DH
```

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

6.1.3 Step 3 – Change Default Password

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

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

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

> For information on the BMC's password policy, refer to section "BMC Password Policy".

For example:

```
[redfish_scripts] $ curl -k -u root:0penBmc -H "Content-Type: application/json" -X PATCH https://<BF-BMC-IP>/
redfish/v1/AccountService/Accounts/root -d '{"Password" : "HelloNvidia3D!"}'
Response:
{
  "@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The request completed successfully.",
      "MessageArgs": [],
      "MessageId": "Base.1.15.0.Success",
      "MessageSeverity": "OK",
      "Resolution": "None"
    }
  ]
}
```

6.1.4 Step 4 – Upgrade BlueField Firmware Components and BSP

Upgrade the BlueField firmware components (i.e., ATF, UEFI, NIC-firmware, DPU BMC, and ERoT) and the BSP using the BFB image according to the following instructions.

> Make sure to download the latest DOCA image (BFB file) available from the NVIDIA DOCA Downloader.

Refer to section "BFB Installation" for a detailed procedure.

6.1.5 Step 5 – Verify Software Component Versions

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

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

```
{
  "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory",
  "@odata.type": "#SoftwareInventoryCollection.SoftwareInventoryCollection",
  "Members": [
    {
      "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/BMC_Firmware"
    },
    {
```

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


Response example for `DPU_ATF`:


```
> curl -k -u root:<password> -H "Content-Type: application/octet-stream" -X GET https://<BF-BMC-IP>/
redfish/v1/UpdateService/FirmwareInventory/DPU_ATF
{
  "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_ATF",
  "@odata.type": "#SoftwareInventory.v1_4_0.SoftwareInventory",
  "Description": "Host image",
  "Id": "DPU_ATF",
  "Members@odata.count": 1,
  "Name": " "Software Inventory",
  "RelatedItem": [
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/Bios"
    }
  ],
  "SoftwareId": "",
  "Status": {
    "Health": "OK",
    "HealthRollup": "OK",
    "State": "OK",
  },
  "Updateable": true,
  "Version": "v2.2(release):4.0.2-33-gd9f4ad5"
```


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

2. If the versions are not as expected, upgrade as needed:
    a. Download the latest DOCA (BFB file) versions from the downloader at the bottom of the DOCA product page.
    b. DOCA (BFB) upgrade options (upgrading UEFI, ATF, Arm OS, NIC firmware components):
        - Recommended—BFB upgrade from remote management controller using Redfish `UpdateService` schema over 1GbE to BlueField BMC:

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

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

6.1.6 Step 6 – Relate BlueField to BlueField BMC and NIC Data Ports on Same Machine

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

```
curl -k -u root:<password> -H 'Content-Type: application/json' -X GET https://<BF-BMC-IP>/redfish/v1/
Managers/Bluefield_BMC/EthernetInterfaces/eth0
 {
   "@odata.id": "/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0",
   "@odata.type": "#EthernetInterface.v1_6_0.EthernetInterface",
   "DHCPv4": {
     "DHCPEnabled": true,
     "UseDNSServers": true,
     "UseDomainName": true,
     "UseNTPServers": true
   },
   "DHCPv6": {
     "OperatingMode": "Stateful",
     "UseDNSServers": true,
     "UseDomainName": true,
     "UseNTPServers": true
   },
   "Description": "Management Network Interface",
   "FQDN": "dpu-bmc",
   "HostName": "BlueField-bmc",
   "IPv4Addresses": [
     {
       "Address": "10.237.40.179",
       "AddressOrigin": "DHCP",
       "Gateway": "0.0.0.0",
       "SubnetMask": "255.255.0.0"
     }
   ],
   "IPv4StaticAddresses": [],
   "IPv6AddressPolicyTable": [],
   "IPv6Addresses": [
     {
       "Address": "fdfd:fdfd:10:237:966d:aeff:fe17:9f5f",
       "AddressOrigin": "DHCPv6",
       "AddressState": null,
       "PrefixLength": 64
     },
     {
       "Address": "fe80::966d:aeff:fe17:9f5f",
       "AddressOrigin": "LinkLocal",
       "AddressState": null,
       "PrefixLength": 64
     }
   ],
   "IPv6DefaultGateway": "fe80::445b:ed80:5f97:8900",
   "IPv6StaticAddresses": [],
   "Id": "eth0",
   "InterfaceEnabled": true,
   "LinkStatus": "LinkUp",
   "MACAddress": "94:6d:ae:17:9f:5f",
   "MTUSize": 1500,
   "Name": "Manager Ethernet Interface",
   "NameServers": [
     "fdfd:fdfd:7:77:250:56ff:fe8b:e4f9"
   ],
   "SpeedMbps": 0,
   "StaticNameServers": [],
   "Status": {
     "Health": "OK",
     "HealthRollup": "OK",
     "State": "Enabled"
   },
   "VLANs": {
     "@odata.id": "/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0/VLANs"
   }
 }
```

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

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

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


6.1.7 Step 7 – Change Mode of Operation to Zero-trust Mode

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

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

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

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

> ⚠ Changing host privilege level requires BlueField DPU reset and host power cycle for the change to take effect.

> ℹ For more information on BlueField modes of operation, refer to this page.

6.1.8 Step 8 – (Optional) Disable Secure Boot

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

To disable secure boot using the Redfish `SecureBoot` schema over 1GbE to BlueField BMC, follow the command in section "Setting Secure Boot State".

> ℹ For more information on user management, review this page.

6.2 NIC Mode Installation

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

> ⚠ NIC mode is the default mode for BlueField SuperNICs, while BlueField DPUs are shipped with DPU mode as their default. To switch between the modes, see NVIDIA BlueField Modes of Operation. To check which mode your BlueField is currently running, see Common Configurations.

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

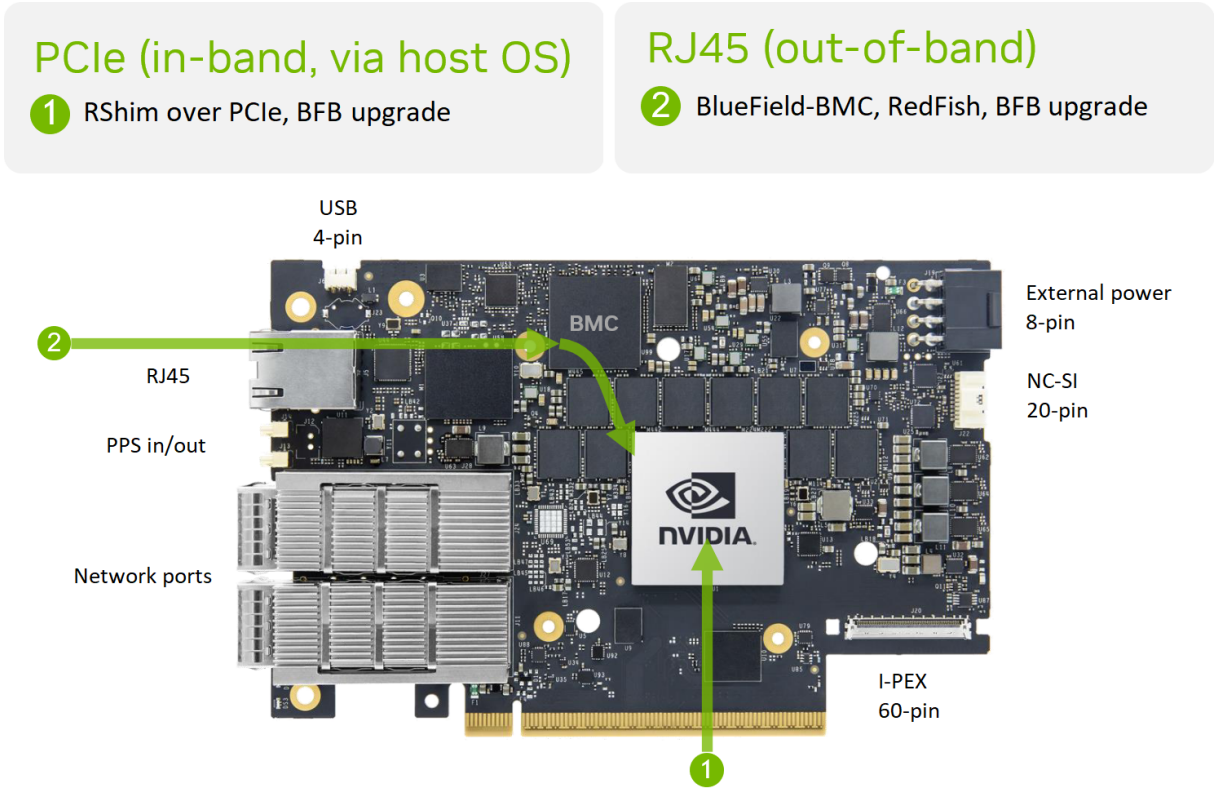
6.2.1 Upgrade BlueField Firmware Components and BSP Using BFB Image

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

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

This can be performed using one of the following methods:

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



6.2.2 Changing UEFI and BMC Password Using bf.cfg

- To change the UEFI password, add the current UEFI password under parameter `UEFI_PASSWORD` and define the new UEFI password under `NEW_UEFI_PASSWORD` inside the `bf.cfg` configuration file.
- To change the BMC root password, add the current BMC root password under parameter `BMC_PASSWORD` and define the new BMC root password under `NEW_BMC_PASSWORD` inside the `bf.cfg` configuration file.

6.2.3 Change Mode of Operation to Zero-trust Mode

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

Set RShim State Command Format

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

Set RShim State Command Parameters

Field	Bytes	Offset in NC-SI Command	Description
Host_RT_Access_State	1	27	RShim state: • 0 – Enabled • 1 – Locked • Other – reserved

7 BlueField Management

7.1 Management in BlueField BMC

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

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

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

Managing the BlueField using its BMC is detailed hereafter.

7.1.1 Remote Management Using Redfish Protocol

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

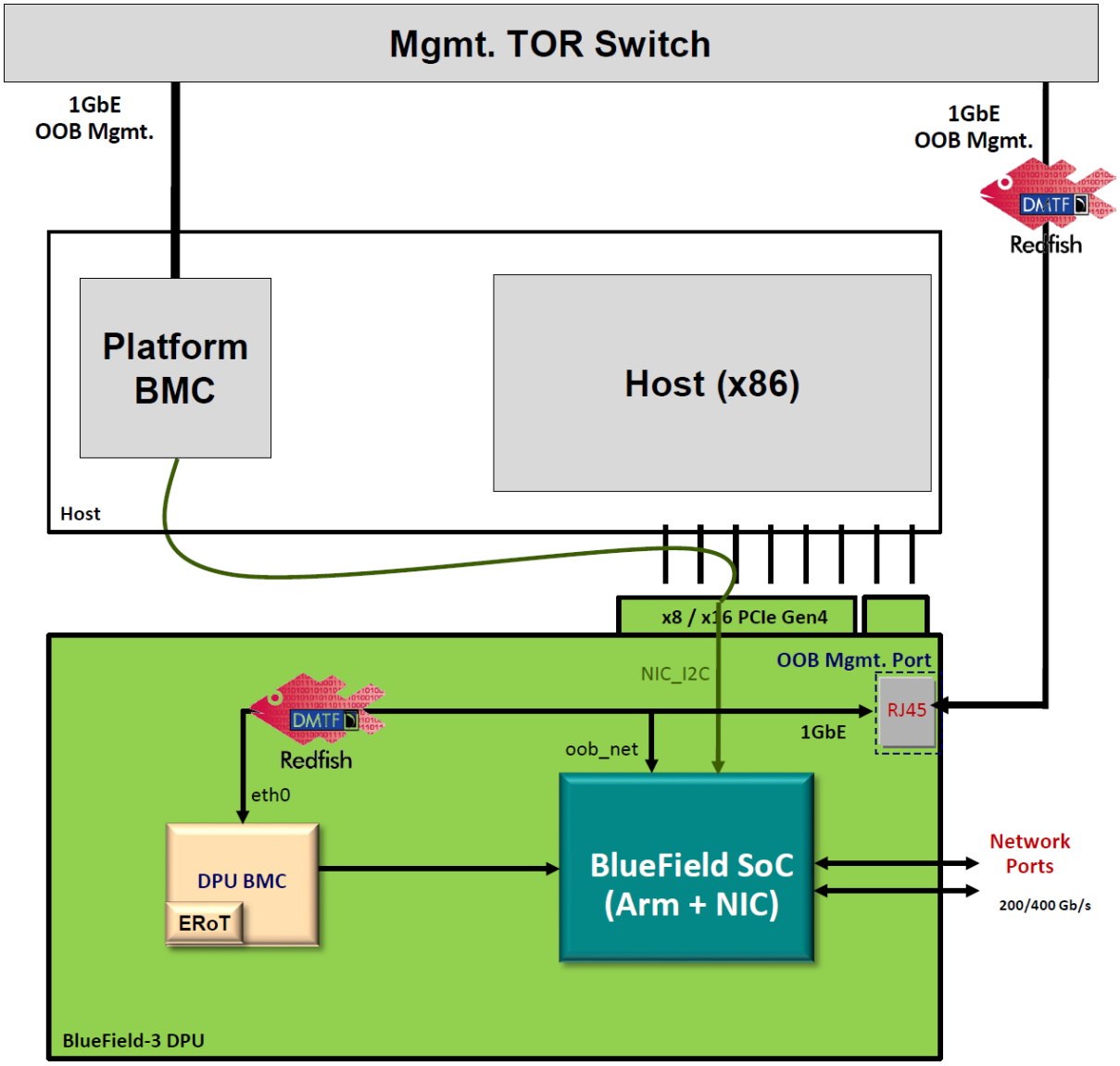
Redfish is supported from the host BMC and BlueField BMC.

Redfish replaces IPMI, providing the following advantages:

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

7.1.2 Management Architecture

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



7.1.3 Management Interfaces

See this page for a detailed description of the interfaces of BlueField-3.

See here page for a detailed description of the interfaces of BlueField-2.

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

Management Interface	Description	Comment
OOB Management Port (1GbE RJ45)	A dedicated, separate Ethernet interface to manage the BlueField from the remote management controller (RMC)	NVIDIA recommends using this interface as the main management interface. Enables managing the BlueField life cycle using the BlueField's BMC. Supports Redfish commands to the BlueField BMC (`eth0`). Recovery flows, monitoring, and configuration operations are all available through this interface. In addition, this physical interface allows users to SSH directly to the BlueField (`oob_net`). IPMI is supported for backward compatibility, but it is recommended to start new deployments with Redfish only.
SMBus (PCIe Golden Fingers)	Enables PLDM/NC-SI over MCTP between the BlueField and the host BMC	Enables the host BMC to monitor the BlueField
PCIe	PCIe interface between the BlueField and host server	Enables the host to recover the BlueField using RShim PCIe physical function (PF) when the host is trusted Unavailable while in zero-trust mode. Use the 1GbE OOB interface instead.

7.1.4 Recommended Management Approach

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

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

7.2 Management Methods

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

The following method are not supported by BlueField SuperNIC.

7.2.1 BlueField Update and Recovery

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

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

Each file format has a different update and recovery method:

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

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

7.2.2 BlueField BMC Update

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

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

Please refer to the "CEC and BMC Firmware Operations" page for more information.

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

7.2.3 BlueField Monitoring and Telemetry

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

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

Please refer to section "Sensor Redfish Commands" for more information.

7.2.4 BlueField and BlueField BMC Reset Control

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

> Please refer to the "Reset Control" page and section "Resetting CEC and BMC Subsystems Using CEC Self-reset Command" for more information.

7.2.5 BlueField UEFI Configuration

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

> Please refer to the "Platform Management Interface" page for more information.

7.2.6 Console Interface

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

> Please refer to the "BIOS Configuration" page for more information.

7.3 BlueField Management Topics

This following high-level topics allow for easy navigation of management options for your BlueField device in DPU mode:

- Common Configurations
- Update and Recovery
- Monitoring
- Network
- Miscellaneous
- Reset Control
- Factory Reset
- DPU BMC SPDM Attestation via Redfish

7.4 Common Configurations

This section contains the following pages:

7.4.1 BlueField Modes of Operation Configuration

This page details the different operational modes for NVIDIA® BlueField® networking platforms (DPU, Zero-trust, and NIC modes) and provides instructions for configuring and switching between them.

7.4.1.1 Introduction

BlueField devices feature the following modes of operation:

Mode of Operation	External Host Trust Level	Description	Default on SKUs	Can be Configured to All Other Modes?
NIC Mode	Host-trusted	The Arm cores of BlueField are inactive, and the device functions as an NVIDIA® ConnectX® network adapter.	SuperNIC SKUs default mode	Yes Broken macro
DPU Mode	Host-trusted	The Arm cores of BlueField are active, and the embedded Arm system runs services that manage the NIC resources and data path.	DPU SKUs default mode	Yes
	Zero Trust (restricted)	The Arm cores of BlueField are active, and the embedded Arm system runs services to manage the NIC resources and data path while enforcing restrictions on the external host (host isolation).	-	Yes

We don't have a way to export this macro.

7.4.1.1.1 NIC Mode

In NIC Mode, BlueField operates as a ConnectX network adapter for the external host. For BlueField-3, the Arm cores are inactive, while for BlueField-2, the Arm cores are active but non-functional.

Operating in NIC Mode on BlueField-3 reduces power consumption, improves network performance, and minimizes the host memory footprint.

BlueField-3 SuperNIC SKUs are shipped in NIC Mode by default.

Multi-host is not supported when BlueField is operating in NIC Mode.

7.4.1.1.2 DPU Mode

In this operation mode, Arm cores active, known also as embedded CPU function ownership (ECPF) mode, is the default mode for the BlueField DPU family of SKUs.

In DPU Mode, the NIC resources and functionality are owned and controlled by the embedded Arm subsystem. All network communication to the host flows through a virtual switch control plane hosted on the Arm cores that manages all networking traffic coming and going from the host.

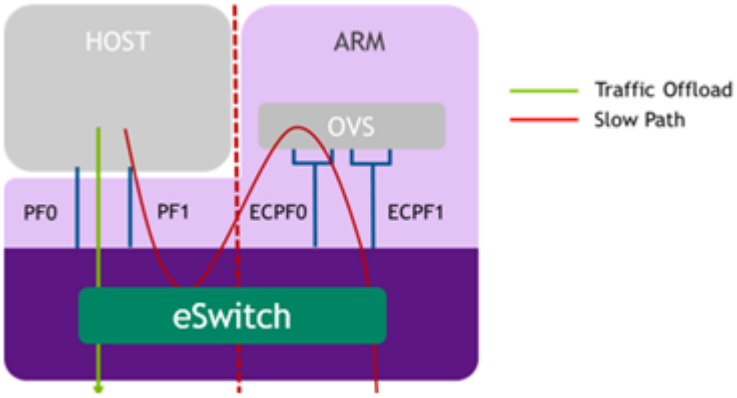
While working in this mode, the BlueField is the trusted function managed by the data center and host administrator for provisioning, management and orchestration (eg. load network drivers, reset an interface, bring an interface up and down, update the firmware, change the mode of operation on BlueField, etc).

⚠ BlueField-2 DPU and BlueField-3 DPU SKUs are shipped in DPU Mode by default.

⚠ Socket Direct is not supported when BlueField is operating in DPU Mode.

7.4.1.1.2.1 DPU Mode Architecture

In DPU Mode, the BlueField DPU provides the host system with access to network functions, while the host's capabilities are restricted and managed by the Arm processor within the BlueField.



Traffic Management and Paths

The Embedded Control and Processing Framework (ECPF) controls the NIC's embedded switch (eswitch). All network traffic between the host interface and the network initially passes through the BlueField's Arm processor via Representors. This path, where traffic is processed by the Arm processor, is referred to as the "slow path".

To improve performance, the Arm processor can define rules in the eswitch through the ECPF, allowing packets to bypass the Arm processor and be processed directly by the eswitch. This is known as the "fast path", which reduces latency and increases throughput by offloading traffic processing from the Arm to the eswitch.

A virtual switch running on the Arm processor may integrate both slow path and fast path functionalities by processing and classifying only the first packet of a new flow. For subsequent packets in the flow, the virtual switch defines eswitch rules, enabling fast path processing for the remainder of the traffic.

Initialization Process

At startup, network access to the host is initially blocked. This restriction remains until the virtual switch running on the Arm processor loads the default out-of-box rules to manage the ECPF on the BlueField. Once these rules are loaded, network traffic to the host is automatically enabled.

> ⚠ The driver on the host system can only be loaded after the driver on the BlueField has been loaded and has completed NIC configuration. Additionally, all Interface Configuration Memory (ICM) is allocated by the ECPF and resides in the BlueField's memory.

7.4.1.1.2.2 InfiniBand in DPU Mode

In DPU Mode, when operating with an InfiniBand network, OpenSM must be executed from the BlueField Arm side rather than the host side. Similarly, InfiniBand management tools such as `sminfo`, `ibdev2netdev`, and `ibnetdiscover` can only be used from the BlueField Arm side and are not accessible from the host side.

7.4.1.1.3 DPU Mode in Zero Trust

Zero Trust, also known as Restricted Mode, is a specialized variation of DPU Mode that enhances security by preventing the host system administrator from accessing BlueField from the host side. Once Zero Trust mode is enabled, the BlueField must be fully controlled by the data center administrator via the Arm cores or the BMC connection, rather than through the host.

This mode enforces security and isolation by restricting the host from performing operations that could compromise BlueField. The following operations can be restricted individually in Zero Trust mode:

- Port ownership – The host cannot assign itself as the port owner
- Hardware counters – The host is denied access to hardware counters
- Tracer functionality – The tracer functionality is blocked
- RShim interface – The RShim interface is disabled
- Firmware flash – firmware flashing from the host is restricted

Zero Trust mode ensures a robust security boundary between the host and BlueField, making it an ideal configuration for environments requiring strict control and isolation.

7.4.1.2 Moving Between Operation Modes

Transitioning between operation modes can be achieved through various configuration interfaces as presented in the following table.

> ⚠ Some configuration interfaces may be locked out when operating in Zero Trust (Restricted) Mode.

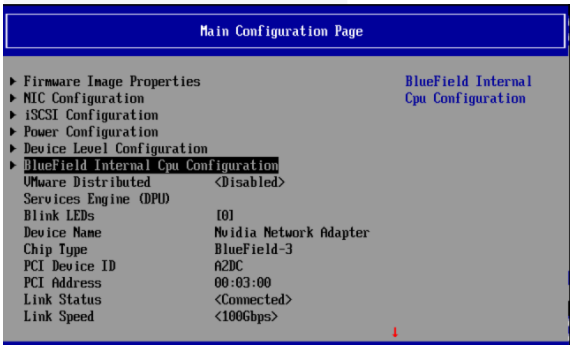
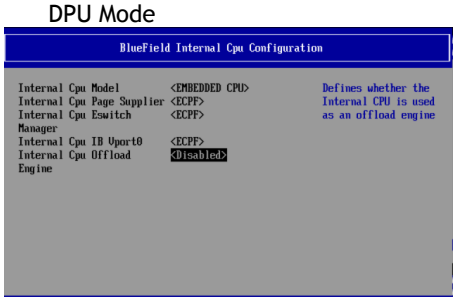
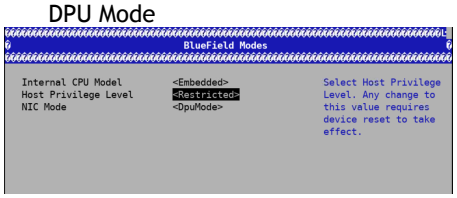
Interface	From	To	Configuration option available by default for this interface?	Configuration option can be locked-out for this interface?
External Host command line	DPU Mode	NIC Mode	Yes	Yes
	NIC Mode	DPU Mode	Yes	Yes
	DPU Mode	DPU Mode with Zero Trust	No	Always blocked
	DPU Mode with Zero Trust	DPU Mode	No	Always blocked
	DPU Mode with Zero Trust	NIC Mode	Not supported	N/A
Host UEFI Menu	DPU Mode	NIC Mode	Yes	Yes
	NIC Mode	DPU Mode	Yes	Yes
	DPU Mode	DPU Mode with Zero Trust	No	Always blocked
	DPU Mode with Zero Trust	DPU Mode	No	Always blocked
	DPU Mode with Zero Trust	NIC Mode	Not supported	N/A
Arm OS command line	DPU Mode	NIC Mode	Yes	No
	NIC Mode	DPU Mode	N/A	N/A
	DPU Mode	DPU Mode with Zero Trust	Yes	No
	DPU Mode with Zero Trust	DPU Mode	Yes	No
	DPU Mode with Zero Trust	NIC Mode	Not supported	N/A
Arm UEFI Menu	DPU Mode	NIC Mode	Yes	No
	NIC Mode	DPU Mode	Yes	No
	DPU Mode	DPU Mode with Zero Trust	No Broken macro	No
	DPU Mode with Zero Trust	DPU Mode	No Broken macro	No

Interface	From	To	Configuration option available by default for this interface?	Configuration option can be locked-out Broken macro for this interface?
	DPU Mode with Zero Trust	NIC Mode	Not supported	N/A
DPU-BMC Redfish	DPU Mode	NIC Mode	Yes	No
	NIC Mode	DPU Mode	Yes	No
	DPU Mode	DPU Mode with Zero Trust	No Broken macro	No
	DPU Mode with Zero Trust	DPU Mode	No Broken macro	No
	DPU Mode with Zero Trust	NIC Mode	Not supported	N/A
Platform-BMC NC-SI OEM commands	DPU Mode	NIC Mode	Yes	No
	NIC Mode	DPU Mode	Yes	No
	DPU Mode	DPU Mode with Zero Trust	No Broken macro	No
	DPU Mode with Zero Trust	DPU Mode	No Broken macro	No
	DPU Mode with Zero Trust	NIC Mode	Not supported	N/A

We don't have a way to export this macro.

7.4.1.2.1 Identifying Which Mode BlueField is Currently Operating In

Interface	Command	Response
Using external host command line	`host> sudo mlxconfig -d /dev/mst/<device> q INTERNAL_CPU_OFFLOAD_ENGINE`	Output format: `RO INTERNAL_CPU_OFFLOAD_ENGINE ENABLED(0)` • `ENABLED(0)` means BlueField is running in DPU Mode • `DISABLED(1)` means BlueField is running in NIC Mode • `RO` is marked for read only, and indicates DPU Mode with Zero Trust

Interface	Command	Response
Using external host UEFI (HII) menu	Navigate to specific device and select `BlueField Internal Cpu Configuration`. 	`Internal Cpu Offload Engine` indicates the mode BlueField is currently operating in: • Disabled - BlueField is operating in NIC Mode • Enabled - BlueField is operating in DPU Mode 
Using Arm UEFI (GUI) menu	Access the Arm UEFI menu by pressing the Esc button twice on the console and navigate to `Device Manager` → `System Configuration` → `BlueField Modes`.	The `NIC Mode` menu indicates the mode BlueField is currently operating in: • `NicMode` - BlueField is operating in NIC Mode • `DpuMode` - BlueField is operating in DPU Mode 
Using DPU BMC Redfish	``` curl -k -u root:'<PASSWORD>' -X GET https://<bmc_ip>/redfish/v1/Systems/Bluefield/Oem/Nvidia ```	Read `Mode` field: ``` { ... "Mode": <"DpuMode"> or <"NicMode"> ... } ``` • `"Mode": NicMode` - BlueField is operating in NIC Mode • `"Mode":DpuMode` - BlueField is operating in DPU Mode

<table>
<tr><th>Interface</th><th>Command</th><th>Response</th></tr>
<tr><td>Using platform BMC NC-SI OEM commands</td><td>

Get Command = 0x13, Parameter = 0x33

<table>
<tr><th>Offset</th><th>31:24</th><th>23:16</th><th>15:8</th><th>7:0</th></tr>
<tr><td>0:15</td><td colspan="4">NC-SI OEM Header (OEM Command)</td></tr>
<tr><td>16:19</td><td colspan="4">Mellanox Manufacture ID (IANA) = 0x8119</td></tr>
<tr><td>20:23</td><td>Command rev=0x00</td><td>MLNX Cmd ID= 0x13</td><td>Parameter=0x33</td><td>Reserved</td></tr>
<tr><td>24:27</td><td colspan="4">Checksum 31:0</td></tr>
</table>

</td><td>

Response format:

<table>
<tr><th>Offset</th><th>31:24</th><th>23:16</th><th>15:8</th><th>7:0</th></tr>
<tr><td>0:15</td><td colspan="4">NC-SI OEM Header (OEM Command)</td></tr>
<tr><td>16:19</td><td colspan="2">Response Code</td><td colspan="2">Reason Code</td></tr>
<tr><td>20:23</td><td colspan="4">Mellanox Manufacture ID (IANA) = 0x8119</td></tr>
<tr><td>24:27</td><td>Command rev=0x00</td><td>MLNX Cmd ID= 0x12</td><td>Parameter=0x33</td><td>Reserved</td></tr>
<tr><td>28:31</td><td colspan="3"></td><td>Offload engine</td></tr>
<tr><td>32:35</td><td colspan="4">Checksum 31:0</td></tr>
</table>

Response field:

<table>
<tr><th>Field</th><th>Size</th><th>Offset in NC-SI Command</th><th>Description</th></tr>
<tr><td>Offload engine</td><td>1 bit</td><td>31.0</td><td>0x0: Enabled - DPU Mode
0x1: Disabled - NIC Mode</td></tr>
</table>

</td></tr>
</table>

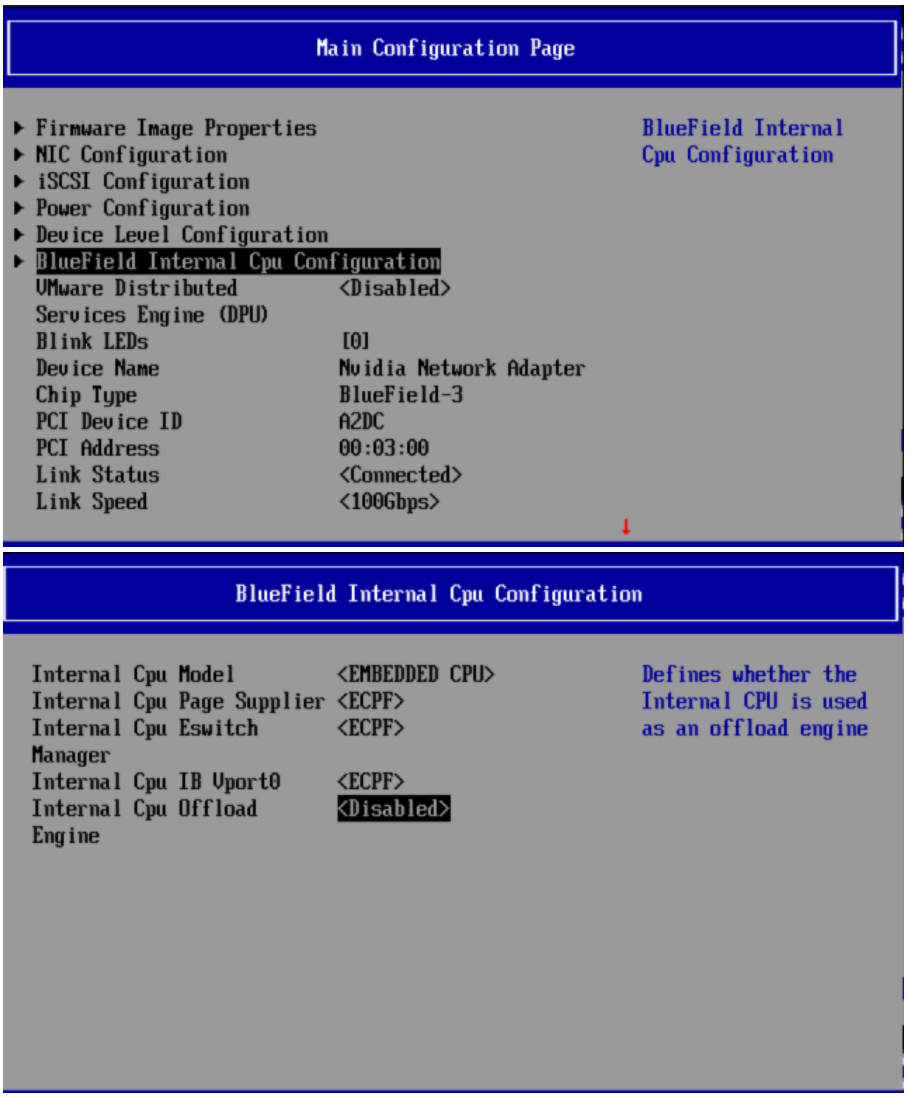
7.4.1.2.2 Changing Mode

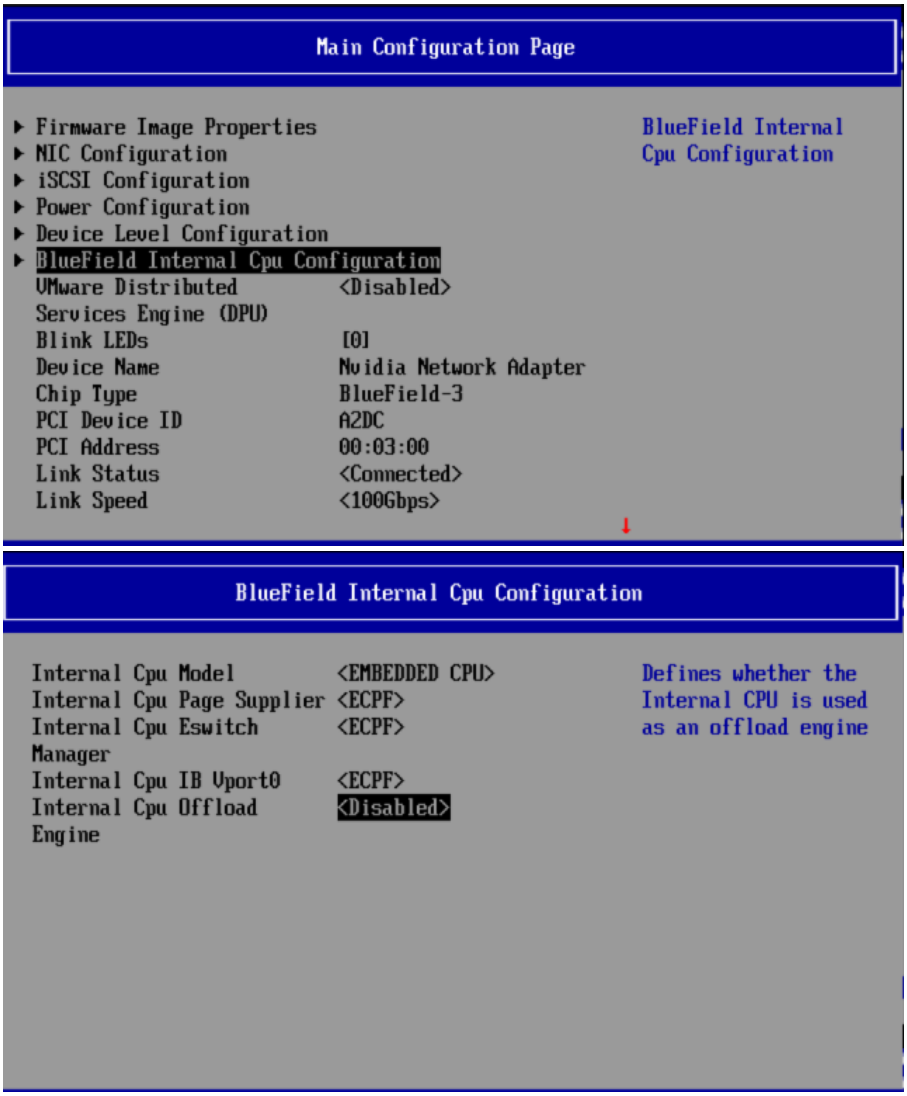
For the configuration to take effect, Arm and NIC components must undergo a reset. Power cycle is recommended.

7.4.1.2.2.1 Using External Host Command Line

From Mode	To Mode	Command
DPU Mode	NIC Mode	For BlueField-3: `host> sudo mlxconfig -d /dev/mst/<device> s INTERNAL_CPU_OFFLOAD_ENGINE=1` For BlueField-2: `host> sudo mlxconfig -d /dev/mst/<device> s INTERNAL_CPU_PAGE_SUPPLIER=1 INTERNAL_CPU_ESWITCH_MANAGER=1 INTERNAL_CPU_IB_VPORT0=1 INTERNAL_CPU_OFFLOAD_ENGINE=1`
NIC Mode	DPU Mode	For BlueField-3: `host> sudo mlxconfig -d /dev/mst/<device> s INTERNAL_CPU_OFFLOAD_ENGINE=0` For BlueField-2: `host> sudo mlxconfig -d /dev/mst/<device> s INTERNAL_CPU_PAGE_SUPPLIER=0 INTERNAL_CPU_ESWITCH_MANAGER=0 INTERNAL_CPU_IB_VPORT0=0 INTERNAL_CPU_OFFLOAD_ENGINE=0`
DPU Mode	DPU Mode with Zero Trust	N/A
DPU Mode with Zero Trust	DPU Mode	N/A
DPU Mode with Zero Trust	NIC Mode	Not supported. Move from DPU Mode with Zero Trust to DPU Mode first, and then from DPU Mode to NIC Mode. • Perform a system-level reset when moving from Zero Trust to DPU Mode before configuring NIC Mode. • Ensure to disable Zero Trust configuration before transitioning to NIC Mode. • Operating in NIC Mode with Zero Trust (Restricted) configuration is not supported and may lead to undefined behavior.

7.4.1.2.2.2 Using External Host UEFI Menu

From Mode	To Mode	Command
DPU Mode	NIC Mode	 Select `BlueField Internal Cpu Configuration`. To enable NIC Mode, set `Internal Cpu Offload Engine` to `Disabled`.

From Mode	To Mode	Command
NIC Mode	DPU Mode	 Select `BlueField Internal Cpu Configuration`. To enable DPU Mode, set `Internal Cpu Offload Engine` to `Enabled`.
DPU Mode	DPU Mode with Zero Trust	N/A
DPU Mode with Zero Trust	DPU Mode	N/A

From Mode	To Mode	Command
DPU Mode with Zero Trust	NIC Mode	Not supported. Move from DPU Mode with Zero Trust to DPU Mode first, and then from DPU Mode to NIC Mode. • Perform a system-level reset when moving from Zero Trust to DPU Mode before configuring NIC Mode. • Ensure to disable Zero Trust configuration before transitioning to NIC Mode. • Operating in NIC Mode with Zero Trust (Restricted) configuration is not supported and may lead to undefined behavior.

7.4.1.2.2.3 Using Arm OS Command Line

From Mode	To Mode	Command
DPU Mode	NIC Mode	For BlueField-3: `bf> sudo mlxconfig -d /dev/mst/<device> s INTERNAL_CPU_OFFLOAD_ENGINE=1` For BlueField-2: `bf> sudo mlxconfig -d /dev/mst/<device> s INTERNAL_CPU_PAGE_SUPPLIER=1 INTERNAL_CPU_ESWITCH_MANAGER=1 INTERNAL_CPU_IB_VPORT0=1 INTERNAL_CPU_OFFLOAD_ENGINE=1`
NIC Mode	DPU Mode	Not Applicable (Arm OS is not available)
DPU Mode	DPU Mode with Zero Trust	`bf> $ sudo mlxprivhost -d /dev/mst/<device> r --disable_rshim --disable_tracer --disable_counter_rd --disable_port_owner`
DPU Mode with Zero Trust	DPU Mode	`bf> $ sudo mlxprivhost -d /dev/mst/<device> p`
DPU Mode with Zero Trust	NIC Mode	Not supported. Move from DPU Mode with Zero Trust to DPU Mode first, and then from DPU Mode to NIC Mode. • Perform a system-level reset when moving from Zero Trust to DPU Mode before configuring NIC Mode. • Ensure to disable Zero Trust configuration before transitioning to NIC Mode. • Operating in NIC Mode with Zero Trust (Restricted) configuration is not supported and may lead to undefined behavior.

7.4.1.2.2.4 Using Arm UEFI Menu

From Mode	To Mode	Command
DPU Mode	NIC Mode	1. Access the Arm UEFI menu by pressing the Esc button twice on the console. 2. Select `Device Manager` → `System Configuration` → `BlueField Modes`. 3. Set the `NIC Mode` option to `NicMode` to enable NIC Mode.  4. Exit `BlueField Modes` and `System Configuration` and make sure to save the settings. 5. Exit the UEFI setup using the "reset" option. The configuration is not yet applied and BlueField is expected to boot in DPU Mode. 6. Issue power cycle to apply new configuration.
NIC Mode	DPU Mode	1. Access the Arm UEFI menu by pressing the Esc button twice on the console. 2. Select `Device Manager` → `System Configuration` → `BlueField Modes`. 3. Set the `NIC Mode` option to `DpuMode` to enable DPU Mode.  4. Exit `BlueField Modes` and `System Configuration` and make sure to save the settings. 5. Exit the UEFI setup using the "reset" option. The configuration is not yet applied and BlueField is expected to boot in NIC Mode. 6. Issue power cycle to apply new configuration.
DPU Mode	DPU Mode with Zero Trust	Roadmap
DPU Mode with Zero Trust	DPU Mode	Roadmap
DPU Mode with Zero Trust	NIC Mode	Not supported. Move from DPU Mode with Zero Trust to DPU Mode first, and then from DPU Mode to NIC Mode. • Perform a system-level reset when moving from Zero Trust to DPU Mode before configuring NIC Mode. • Ensure to disable Zero Trust configuration before transitioning to NIC Mode. • Operating in NIC Mode with Zero Trust (Restricted) configuration is not supported and may lead to undefined behavior.

7.4.1.2.2.5 Using DPU-BMC Redfish

From Mode	To Mode	Command
DPU Mode	NIC Mode	`curl -k -u root:'password' -H 'content-type: application/json' -d '{"Mode": "NicMode"}' -X POST https://bmc_ip/redfish/v1/Systems/Bluefield/Oem/Nvidia/Actions/Mode.Set` Two consecutive Arm reboots are required to apply configuration.
NIC Mode	DPU Mode	`curl -k -u root:'password' -H 'content-type: application/json' -d '{"Mode": "DpuMode"}' -X POST https://bmc_ip/redfish/v1/Systems/Bluefield/Oem/Nvidia/Actions/Mode.Set` Two consecutive Arm reboots are required to apply configuration.
DPU Mode	DPU Mode with Zero Trust	`curl -k -u root:'password' -H 'content-type: application/json' -d '{"PrivilegeMode": "Restricted"}' -X PATCH https://<bmc_ip>/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/Oem/Nvidia/HostPrivilegeConfig/Settings` Power cycle is required to apply configuration.
DPU Mode with Zero Trust	DPU Mode	`curl -k -u root:'password' -H 'content-type: application/json' -d '{"PrivilegeMode": "Privileged"}' -X PATCH https://<bmc_ip>/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/Oem/Nvidia/HostPrivilegeConfig/Settings` Power cycle is required to apply configuration.
DPU Mode with Zero Trust	NIC Mode	Not supported. Move from DPU Mode with Zero Trust to DPU Mode first, and then from DPU Mode to NIC Mode. • Perform a system-level reset when moving from Zero Trust to DPU Mode before configuring NIC Mode. • Ensure to disable Zero Trust configuration before transitioning to NIC Mode. • Operating in NIC Mode with Zero Trust (Restricted) configuration is not supported and may lead to undefined behavior.

7.4.1.2.2.6 Using Platform-BMC NC-SI OEM Commands

<table>
<tr><th>From Mode</th><th>To Mode</th><th>Command</th></tr>
<tr><td>DPU Mode</td><td>NIC Mode</td><td>

Command = 0x12, Parameter = 0x33

<table>
<tr><th>Offset</th><th>31:24</th><th>23:16</th><th>15:8</th><th>7:0</th></tr>
<tr><td>0:15</td><td colspan="4">NC-SI OEM Header (OEM Command)</td></tr>
<tr><td>16:19</td><td colspan="4">Mellanox Manufacture ID (IANA) = 0x8119</td></tr>
<tr><td>20:23</td><td>Command rev=0x00</td><td>MLNX Cmd ID= 0x12</td><td>Parameter =0x33</td><td>Reserved</td></tr>
<tr><td>24:27</td><td colspan="2"></td><td></td><td>Offload engine</td></tr>
<tr><td>28:31</td><td colspan="4">Checksum 31:0</td></tr>
</table>

<table>
<tr><th>Field</th><th>Size</th><th>Offset in NC-SI Command</th><th>Description</th></tr>
<tr><td>Offload engine</td><td>1 bit</td><td>27.0</td><td>0x0: Enabled – DPU Mode
0x1: Disabled – NIC Mode</td></tr>
</table>

For NIC Mode, set the offload engine bit to 0x1.

</td></tr>
<tr><td>NIC Mode</td><td>DPU Mode</td><td>

Command = 0x12, Parameter = 0x33

<table>
<tr><th>bytes/ bits</th><th>31:24</th><th>23:16</th><th>15:8</th><th>7:0</th></tr>
<tr><td>0:15</td><td colspan="4">NC-SI OEM Header (OEM Command)</td></tr>
<tr><td>16:19</td><td colspan="4">Mellanox Manufacture ID (IANA) = 0x8119</td></tr>
<tr><td>20:23</td><td>Command rev=0x00</td><td>MLNX Cmd ID= 0x12</td><td>Parameter=0x33</td><td>Reserved</td></tr>
<tr><td>24:27</td><td colspan="2"></td><td></td><td>Offload engine</td></tr>
<tr><td>28:31</td><td colspan="4">Checksum 31:0</td></tr>
</table>

<table>
<tr><th>Field</th><th>Size</th><th>Offset in NC-SI Command</th><th>Description</th></tr>
<tr><td>Offload engine</td><td>1 bit</td><td>27.0</td><td>0x0: Enabled - DPU Mode
0x1: Disabled - NIC Mode</td></tr>
</table>

For DPU Mode, set the offload engine bit to 0x0.

</td></tr>
<tr><td>DPU Mode</td><td>DPU Mode with Zero Trust</td><td>Roadmap</td></tr>
</table>

From Mode	To Mode	Command
DPU Mode with Zero Trust	DPU Mode	Roadmap
DPU Mode with Zero Trust	NIC Mode	Not supported. Move from DPU Mode with Zero Trust to DPU Mode first, and then from DPU Mode to NIC Mode. • Perform a system-level reset when moving from Zero Trust to DPU Mode before configuring NIC Mode. • Ensure to disable Zero Trust configuration before transitioning to NIC Mode. • Operating in NIC Mode with Zero Trust (Restricted) configuration is not supported and may lead to undefined behavior.

Separated Host Mode is obsolete for BlueField DPU/SuperNIC SKUs and should not be used, even if it remains visible in some configuration menus or options.

Separated Host Mode is available only for BlueField controller SKUs, where the BlueField Arm OS is the sole CPU/OS in the chassis.

7.4.2 BIOS Secure Boot Configuration

The NVIDIA® BlueField® BMC supports the DMTF Secure Boot schema which enables managing the state of the UEFI Secure Boot through the Redfish interface. This allows clients to set whether UEFI should authenticate the OS image during the boot process.

7.4.2.1 Reading Secure Boot Status

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

Output example:

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/SecureBoot",
  "@odata.type": "#SecureBoot.v1_1_0.SecureBoot",
  "Description": "The UEFI Secure Boot associated with this system.",
  "Id": "SecureBoot",
  "Name": "UEFI Secure Boot",
  "SecureBootCurrentBoot": "Disabled",
  "SecureBootDatabases": {
    "@odata.id": "/redfish/v1/Systems/Bluefield/SecureBoot/SecureBootDatabases"
  },
  "SecureBootEnable": false,
  "SecureBootMode": "SetupMode"
}
```

7.4.2.2 Setting Secure Boot State

The following command enables UEFI Secure Boot through the Redfish interface:

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

The following command disables UEFI Secure Boot through the Redfish interface:

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

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

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

- From the BlueField BMC using Redfish:

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

- From RShim:

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

- From the BlueField Arm OS:

  ```
  reboot
  ```

7.4.2.3 Secure Boot Database Support

The following operations may be performed using Redfish commands. For each operation, a corresponding task is generated within the BMC's Redfish Task Service. During the subsequent BlueField reboot, the UEFI checks for any pending secure boot tasks and executes them in the order of their ascending task ID numbers. After completion, the UEFI then updates the task state to reflect the relevant status.

- To read UEFI Secure boot databases:

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

  Output example:

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/SecureBoot/SecureBootDatabases",
  "@odata.type": "#SecureBootDatabaseCollection.SecureBootDatabaseCollection",
  "Members": [
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/SecureBoot/SecureBootDatabases/KEK"
    },
    ..
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/SecureBoot/SecureBootDatabases/PK"
    },
..
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/SecureBoot/SecureBootDatabases/db"
    },
..
..
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/SecureBoot/SecureBootDatabases/dbx"
    },
..
  ],
  "Members@odata.count": 10,
  "Name": "UEFI SecureBoot Database Collection"
}
```


- To add a certificate to the UEFI `db`:

> ⚠ The following certificate is an example only. The content of the PEM file is copied into the `curl` POST command; `\n` must be used to replace EOL characters. For illustration purposes here's the original content of the PEM certificate file.
>
> ```
> -----BEGIN CERTIFICATE-----
> MIIDbTCCAlWgAwIBAgIUO2MdJt2cTCGr0eO4PiBV5Uk0b/IwDQYJKoZIhvcNAQEL
> BQAwVjELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAk5DMRAwDgYDVQQHEwdSYWxlaWdo
> MQ8wDQYDVQQKEwZMZW5vdm8xFzAVBgNVBAMTDkxlbm92byBVRUZJIERCMB4XDTE3
> MDMxNTIxMTYzNFoXDTQxMDMxNTIxMTYzNFowVjELMAkGA1UEBhMCVVMxCzAJBgNV
> BAgTAk5DMRAwDgYDVQQHEwdSYWxlaWdoMQ8wDQYDVQQKEwZMZW5vdm8xFzAVBgNV
> BAMTDkxlbm92byBVRUZJIERCMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKC
> AQEA1ezmdoBlk5yDVuXw8F774CvllhMz9bT0/TpH3kmRpPAizSRDzbHLUuBMC6SE
> W4zjdLxTqOlwZt6UUcWxlyzKcoDPe43cE6YH1kM/rscvm3AaVL+4GcyGgS9DL6qe
> QFHWER25xCTokMsCdKB42Ty7hWW5FBPepgAS+GDfqQfb/4hoonIlen5X+dqVIsE/
> RM1DIVBUiIbJdgERYeoGjY/Rh4A1VWl6ErzyzokYnf63JjSFR2kVV0apbr4ISOTM
> 7qBd1RNHwQrCAryRADX1XGvRZURzwQdEXfOqZOkVjNKr1fD761qvPE8TQWWJ9Q8r
> mciMocIXqoqWKPAkgbMwKmcsFQIDAQABozMwMTAPBgNVHRMBAf8EBTADAQH/MA4G
> A1UdDwEB/wQEAwICBDAOBgNVHQ8Baf8EBAMCB4AwDQYJKoZIhvcNAQELBQADggEB
> AJ2U0UjB+sxF/HE5sY56vJbdFIT18o0Yf7XJImL0VtgpYjfeqiE768G2uTUbzCKy
> hDOps3+4w4p8FUSO6StzCz6UuUyxlUjQzpkxZ97Ouq1sGhjy7dZybTEByJD6LpI7
> l1EpJSfBiwxTdm7svJoABKs8Hs7e9f3XX5PK76SxllMbDaxAm7UvCppYBBR1SeyC
> gWt3rGRiO3W6pfd07ioCD03kgGzYNOZeU2S+maE1Xt4kUoYs3HxyrhJGfN26gM8h
> 4w5LfCKrlxi+3KMf+vXxEBfGYBvjwcA7KCW92GnUQGVjZbEGs6EaTBx7i9gA2+te
> oWS/500qiwNRp2xqdBxg1d0=
> -----END CERTIFICATE-----
> ```


```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X POST https://<bmc_ip>/redfish/v1/
Systems/Bluefield/SecureBoot/SecureBootDatabases/db/Certificates -d\
 '{"CertificateString": "-----BEGIN CERTIFICATE-----\nMIIDbTCCAlWgAwIBAgIUO2MdJt2cTCGr0eO4PiBV5Uk0b/
IwDQYJKoZIhvcNAQEL\nBQAwVjELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAk5DMRAwDgYDVQQHEwdSYWxlaWdo\nMQ8wDQYDVQQKEwZMZW5vd
m8xFzAVBgNVBAMTDkxlbm92byBVRUZJIERCMB4XDTE3\nMDMxNTIxMTYzNFoXDTQxMDMxNTIxMTYzNFowVjELMAkGA1UEBhMCVVMxCzAJBg
NV\nBAgTAk5DMRAwDgYDVQQHEwdSYWxlaWdoMQ8wDQYDVQQKEwZMZW5vdm8xFzAVBgNV\nBAMTDkxlbm92byBVRUZJIERCMIIBIjANBgkqh
kiG9w0BAQEFAAOCAQ8AMIIBCgKC\nAQEA1ezmdoBlk5yDVuXw8F774CvllhMz9bT0/
TpH3kmRpPAizSRDzbHLUuBMC6SE\nW4zjdLxTqOlwZt6UUcWxlyzKcoDPe43cE6YH1kM/
rscvm3AaVL+4GcyGgS9DL6qe\nQFHWER25xCTokMsCdKB42Ty7hWW5FBPepgAS+GDfqQfb/4hoonIlen5X+dqVIsE/
\nRM1DIVBUiIbJdgERYeoGjY/
Rh4A1VWl6ErzyzokYnf63JjSFR2kVV0apbr4ISOTM\n7qBd1RNHwQrCAryRADX1XGvRZURzwQdEXfOqZOkVjNKr1fD761qvPE8TQWWJ9Q8r
\nmciMocIXqoqWKPAkgbMwKmcsFQIDAQABozMwMTAPBgNVHRMBAf8EBTADAQH/MA4G\nA1UdDwEB/
wQEAwICBDAOBgNVHQ8Baf8EBAMCB4AwDQYJKoZIhvcNAQELBQADggEB\nAJ2U0UjB+sxF/
HE5sY56vJbdFIT18o0Yf7XJImL0VtgpYjfeqiE768G2uTUbzCKy\nhDOps3+4w4p8FUSO6StzCz6UuUyxlUjQzpkxZ97Ouq1sGhjy7dZybT
EByJD6LpI7\nl1EpJSfBiwxTdm7svJoABKs8Hs7e9f3XX5PK76SxllMbDaxAm7UvCppYBBR1SeyC\ngWt3rGRiO3W6pfd07ioCD03kgGzYN
OZeU2S+maE1Xt4kUoYs3HxyrhJGfN26gM8h\n4w5LfCKrlxi+3KMf+vXxEBfGYBvjwcA7KCW92GnUQGVjZbEGs6EaTBx7i9gA2+te\noWS/
500qiwNRp2xqdBxg1d0=\n-----END CERTIFICATE-----","CertificateType": "PEM": "5491316d-9694-4639-b72d-
b8630ffa7dab"}'
```


Output example:


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


- To add a signature to the UEFI `db`:

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X POST https://<bmc_ip>/redfish/v1/
Systems/Bluefield/SecureBoot/SecureBootDatabases/db/Signatures -d\
'{"SignatureString":
"80B4D96931BF0D02FD91A61E19D14F1DA452E66DB2408CA8604D411F92659F0A","SignatureTypeRegistry":
"UEFI","SignatureType": "EFI_CERT_SHA256_GUID": "28d5e212-165b-4ca0-909b-c86b9cee0112"}'
```

Output example:

```
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/1",
  "@odata.type": "#Task.v1_4_3.Task",
  "Id": "1",
  "TaskState": "Pending",
  "TaskStatus": "OK"
}
```

- To delete UEFI `db` certificate #1:

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X DELETE https://<bmc_ip>/redfish/v1/
Systems/Bluefield/SecureBoot/SecureBootDatabases/db/Certificates/1
```

Output example:

```
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/2",
  "@odata.type": "#Task.v1_4_3.Task",
  "Id": "2",
  "TaskState": "Pending",
  "TaskStatus": "OK"
}
```

- To delete all UEFI `db` keys:

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X POST https://<bmc_ip>/redfish/v1/
Systems/Bluefield/SecureBoot/SecureBootDatabases/db/Actions/SecureBootDatabase.ResetKeys
-d'{"ResetKeysType": "DeleteAllKeys"}'
```

Output example:

```
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/3",
  "@odata.type": "#Task.v1_4_3.Task",
  "Id": "3",
  "TaskState": "Pending",
  "TaskStatus": "OK"
}
```

7.4.2.4 Secure Boot Flow Example

The following is an example flow for deleting `PK` certificate using Redfish commands. This command would disable UEFI Secure Boot and revert the system to `Setup Mode`.

1. To reset all `db` keys:

```
root:~# curl -k -u root:'<password>' -H 'Content-Type: application/json' -X POST
https://<bmc_ip>/redfish/v1/Systems/Bluefield/SecureBoot/SecureBootDatabases/PK/Actions/
SecureBootDatabase.ResetKeys -d'{"ResetKeysType": "DeleteAllKeys"}'
```

Output example:

```
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/12",
  "@odata.type": "#Task.v1_4_3.Task",
  "Id": "12",
  "TaskState": "Pending",
  "TaskStatus": "OK"
}
```

Record the returned task ID, in this example the task ID is 12.

2. To read the status of task 12:

```
root:~# curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET https://<bmc_ip>/redfish/
v1/TaskService/Tasks/12
```

Output example:

```
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/12",
  "@odata.type": "#Task.v1_4_3.Task",
  "Id": "12",
  "Messages": [],
  "Name": "Task 12",
  "Payload": {
    "HttpHeaders": [
      "Host: <IP>",
      "User-Agent: curl/7.81.0",
      "Accept: */*",
      "Content-Length: 34"
    ],
    "HttpOperation": "POST",
    "JsonBody": "{\n  \"ResetKeysType\": \"DeleteAllKeys\"\n}",
    "TargetUri": "/redfish/v1/Systems/Bluefield/SecureBoot/SecureBootDatabases/PK/Actions/
SecureBootDatabase.ResetKeys"
  },
  "PercentComplete": 0,
  "StartTime": "2023-09-05T16:47:05+00:00",
  "TaskMonitor": "/redfish/v1/TaskService/Tasks/12/Monitor",
  "TaskState": "Pending",
  "TaskStatus": "OK"
}
```

You can see that `TaskStatus` is `OK` and the `TaskState` is `Pending`. This indicates that the operation has successfully enqueued in the task service and is pending the next BlueField boot.

3. Issue the following graceful reset command to BlueField:

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

Output example:

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

UEFI reads the pending secure boot tasks and executes them.

4. Following BlueField reset, the UEFI updates the status of the operation on the `TaskState` and `TaskStatus` fields. Poll the task and check the values of `TaskState` and `TaskStatus`.

Success	`"TaskState": "Completed",` `"TaskStatus": "OK"`
Failure	`"TaskState": "Exception",` `"TaskStatus": "OK"`

7.4.3 BIOS Configuration

BMC supports configuring the NVIDIA® BlueField®'s BIOS using Redfish commands.

7.4.3.1 BIOS Configuration Schema

The BIOS schema includes properties associated with the BIOS attribute registry, which defines system-specific BIOS attributes and the actions needed to modify BIOS settings. If the `@Redfish.Settings` term is present in this resource, a client can use it to request changes to the BIOS settings by updating the resource identified by the `@Redfish.Settings` annotation.

7.4.3.1.1 Getting BIOS Attributes List

> After running factory reset, the BIOS configuration attributes list is updated only after rebooting the BlueField as the list gets its values from UEFI as BlueField is booting and Redfish is enabled.

```
curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/Registries/BiosAttributeRegistry/
BiosAttributeRegistry
```

Output example:

> In the following output, there is only one BIOS attribute, `UefiPassword`.


```
{
  "@Redfish.Settings": {
    "@odata.type": "#Settings.v1_3_5.Settings",
    "SettingsObject": {
      "@odata.id": "/redfish/v1/Systems/Bluefield/Bios/Settings"
    }
  },
  "@odata.id": "/redfish/v1/Systems/Bluefield/Bios",
  "@odata.type": "#Bios.v1_2_0.Bios",
  "Actions": {
    "#Bios.ChangePassword": {
      "target": "/redfish/v1/Systems/Bluefield/Bios/Actions/Bios.ChangePassword"
    },
    "#Bios.ResetBios": {
      "target": "/redfish/v1/Systems/Bluefield/Bios/Actions/Bios.ResetBios"
    }
  },
  "Attributes": {
    "BootPartitionProtection": false,
    "CeThreshold": 5000,
    "CurrentUefiPassword": "",
    "DateTime": "2024-10-17T19:47:04Z",
    "DefaultPasswordPolicy": true,
    "DisableHEST": false,
    "DisableI2c1": false,
    "DisablePCIe": false,
    "DisableSPMI": false,
    "DisableTMFF": false,
    "EmmcWipe": false,
    "Enable2ndeMMC": false,
    "EnableDdr5600": false,
    "EnableOPTEE": false,
    "EnableSMMU": true,
    "FieldMode": false,
    "ForcePxeRetryDisable": false,
    "HostPrivilegeLevel": "Restricted",
    "InternalCPUModel": "Embedded",
    "L3CachePartitionLevel": 0,
    "LegacyPasswordEnable": false,
    "NicMode": "DpuMode",
    "NvmeWipe": false,
    "OsArgs": "",
    "ResetEfiVars": false,
    "SPCR_UART": "Disabled",
    "UefiArgs": "",
    "UefiPassword": ""
```

```
  },
  "Description": "BIOS Configuration Service",
  "Id": "BIOS",
  "Links": {
    "SoftwareImages": [
      {
        "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_ATF"
      },
      {
        "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_BOARD"
      },
      {
        "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_BSP"
      },
      {
        "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_NIC"
      },
      {
        "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_NODE"
      },
      {
        "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_OFED"
      },
      {
        "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_OS"
      },
      {
        "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_SYS_IMAGE"
      },
      {
        "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_UEFI"
      }
    ],
    "SoftwareImages@odata.count": 9
  },
  "Name": "BIOS Configuration",
  "ResetBiosToDefaultsPending": false
}
```


For information on each of the attributes listed in output, please refer to section "System Configuration" in the UEFI Menu page of the *NVIDIA BlueField BSP* manual.

7.4.3.1.2 Getting Current BIOS Attributes Value

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

Output example:

The current value of `UefiPassword` is an empty string.


```
{
 "@Redfish.Settings": {
   "@odata.type": "#Settings.v1_3_5.Settings",
   "SettingsObject": {
     "@odata.id": "/redfish/v1/Systems/Bluefield/Bios/Settings"
   }
 },
 "@odata.id": "/redfish/v1/Systems/Bluefield/Bios",
 "@odata.type": "#Bios.v1_2_0.Bios",
 ...
 "Attributes": {
   "UefiPassword": ""
 },
 "Description": "BIOS Configuration Service",
 "Id": "BIOS",
 "Name": "BIOS Configuration",
 ...
}
```


7.4.3.1.3 Changing BIOS Attributes Value

Follow this command template to request changing BIOS attribute values:

```
curl -k -u root:'<password>' -X PATCH https://<bmc_ip>/redfish/v1/Systems/Bluefield/Bios/Settings -d '{Attributes:
{<attribute-name> : <attribute-value>}}'
```

At the next boot cycle of the BlueField, the UEFI changes the requested attribute if the requested value is valid.

7.4.3.1.4 Getting Pending BIOS Attribute Values

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

Pending values are a list of values that that user has requested to change. The list of pending values is purged once the UEFI changes the pending attributes.

Output example:

> `UefiPassword` appears in the pending attributes list.

```
{
 "@odata.id": "/redfish/v1/Systems/Bluefield/Bios/Settings",
 "@odata.type": "#Bios.v1_2_0.Bios",
 "Attributes": {
   "UefiPassword": "NewPassword123"
 },
 "Description": "BIOS Settings",
 "Id": "BIOS_Settings",
 "Name": "BIOS Configuration"
}
```

> The active BIOS attribute list is updated only after the UEFI approves the changes during the next reboot cycle.

7.4.3.1.5 BIOS Configuration Examples

7.4.3.1.5.1 Changing Default UEFI Password Using Redfish

1. Look for the "Attributes" property to make sure the UEFI version being used has all the necessary attributes. See section "Get BIOS Attributes List" for instructions.
2. Perform PATCH to BIOS pending settings URI as follows:

   ```
   curl -k -u root:<password> -X PATCH -H "Content-Type: application/json" https://<bmc_ip>/redfish/v1/
   Systems/Bluefield/Bios/Settings -d '{"Attributes":
   {"CurrentUefiPassword":"CurrentPassword","UefiPassword":"NewPassword321"}}'
   ```

3. Reboot BlueField using the Redfish System schema over 1GbE OOB to the BlueField BMC. See section "Reset Control" for instructions.
4. If `CurrentUefiPassword` is correct, then the UEFI password is updated during the UEFI Redfish phase of the boot.

7.4.3.2 BIOS CA Certificates

7.4.3.2.1 Viewing Currently Installed BIOS CA Certificates

The certificates installed on the UEFI may differ from the certificate presented on the BlueField BMC. This discrepancy arises from a distinct certificate validation processed implemented in the UEFI and BlueField BMC.

1. Trigger the following GET request to view the content of the system's Truststore:

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

For example, the following is the response when there are two certificates installed:

```
{
  "@Redfish.SupportedCertificates": [
    "PEM"
  ],
  "@odata.id": "/redfish/v1/Systems/Bluefield/Oem/Nvidia/Truststore/Certificates",
  "@odata.type": "#CertificateCollection.CertificateCollection",
  "Members": [
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/Oem/Nvidia/Truststore/Certificates/1"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/Oem/Nvidia/Truststore/Certificates/2"
    }
  ],
  "Members@odata.count": 2,
  "Name": "TruststoreBios Certificate Collection"
}
```

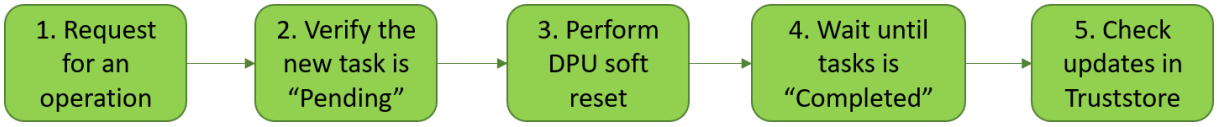
2. Trigger the following GET request to view the details of a specific certificate:

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

For example:

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/Boot/Certificates/1",
  "@odata.type": "#Certificate.v1_7_0.Certificate",
  "CertificateString": "<cert_str>",
  "CertificateType": "PEM",
  "Id": "1",
  "Issuer": {
    "City": "Santa Clara",
    "CommonName": "Kg639IcpJtYMRzvh.nvidia",
    "Country": "US",
    "Organization": "NVIDIA",
    "OrganizationalUnit": "NBU",
    "State": "California"
  },
  "KeyUsage": [
    "CRLSigning"
  ],
  "Name": "TruststoreBios Certificate",
  "Subject": {
    "City": "Santa Clara",
    "CommonName": "Kg639IcpJtYMRzvh.nvidia",
    "Country": "US",
    "Organization": "NVIDIA",
    "OrganizationalUnit": "NBU",
    "State": "California"
  },
  "UefiSignatureOwner": "<UEFI_Owner>",
  "ValidNotAfter": "2043-01-01T00:00:00+00:00",
  "ValidNotBefore": "2023-01-01T00:00:00+00:00"
}
```

7.4.3.2.2 BIOS CA Certificates Collection Operations



1. Request for an operation:
    - To install a certificate, trigger the following POST request which contains the certificate string and type in JSON format:

      > ⚠ The BMC certificate must be replaced with a CA signed certificate before installing new CA certificates, and after BMC factory reset. See section "Example for CSR Generation, Certificate Creation and Replacement" for instructions.

      > ⛔ If an invalid certificate is installed, the BMC rejects it and does not display it. However, it is still accepted by the UEFI and it must be deleted manually through the UEFI menu.

      ```
      curl -k -u root:<password> -X POST https://<bmc_ip>/redfish/v1/Systems/Bluefield/Oem/Nvidia/
      Truststore/Certificates -d @CAcert.json
      ```

      The content of `CAcert.json` must be `{"CertificateString": "<cert_string>", "CertificateType": "<cert_type>"}`. Where:

        - `cert_string` - certification string which starts with `-----BEGIN CERTIFICATE-----\n` and ends with `-----END CERTIFICATE-----\n`.

          > ⚠ The "\n" at the end are mandatory.

        - `cert_type` - certification type. Only "PEM" is supported.

    - To delete a CA certificate, trigger the following `DELETE` request with the CA certificate URI that should be deleted, this only delete it from the BMC trust store:

      ```
      curl -k -u root:<password> -H "Content-Type: application/json" -X DELETE https://<bmc_ip>/redfish/
      v1/Systems/Bluefield/Oem/Nvidia/Truststore/Certificates/<cert_num>
      ```

    - To reset all certificates in the Truststore, trigger the following `TruststoreCertificates.ResetKeys` action with the `DeleteAllKeys` option:

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

2. Verify the new task is `Pending`:
   The responses of these requests indicate the creation of a new task for the UEFI:

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

3. Perform BlueField soft reset in order for the UEFI to start handling the pending tasks:

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

4. Wait until task is `Completed`.
   The task details and status can be checked using the following `GET` request:

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

   The task status can be either:
   - Pending – Initial state.

```
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/<task_id>",
  "@odata.type": "#Task.v1_4_3.Task",
  ...
  "PercentComplete": 0,
  ...
  "TaskState": "Pending",
  "TaskStatus": "OK"
}
```

     If the task remains in a "Pending" state after BlueField reset completes, please check the UEFI-BMC communication.
     If a communication failure occurs, consider either:
     - Replacing the BMC certificate with one signed by a CA whose certificate was installed and resetting BlueField
     - Removing all CA certificates from the UEFI menu
   - Completed – Finished state.

```
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/<task_id>",
  "@odata.type": "#Task.v1_4_3.Task",
  ...
  "PercentComplete": 100,
  ...
  "TaskState": "Completed",
  "TaskStatus": "OK"
}
```

   - Exception – Failure state.

```
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/<task_id>",
  "@odata.type": "#Task.v1_4_3.Task",
  ...
  "PercentComplete": 0,
  ...
  "TaskState": "Exception",
  "TaskStatus": "OK"
}
```

     In this case, verify that the certificate is valid.

5. Check updates in Truststore. See section "Viewing Currently Installed BIOS CA Certificates" for details.

7.4.3.3 BIOS Attributes

For a full list of the BIOS attributes, please refer to the "Redfish" section of the NVIDIA BlueField BSP documentation.

7.4.3.4 BIOS Debug Mode

BIOS debug mode allows users to view the UEFI debug logs when the BlueField Arm OS is booting up.

- To enable the logs, run:

  ```
  ipmitool raw 0x3e 0x24 0x1
  ```

  Returns 01 if successful.
- To disable debug mode, run:

  ```
  ipmitool raw 0x3e 0x24 0x0
  ```

  Returns 00 if successful.
- To query the current applied mode, run:

  ```
  ipmitool raw 0x3e 0x24 0x2
  ```

  Returns:
  - 00 – Normal (default) mode
  - 01 – Debug mode

After setting your desired mode, reset the BlueField Arm OS to view the logs.

Power cycling the system or hard resetting the BlueField SoC resets the BIOS mode value back to its default normal mode.

7.5 Update and Recovery

This section contains the following pages:

- System Inventory
- Deploying Software Using BFB
- Boot Configuration

7.5.1 System Inventory

The Redfish `FirmwareInventory` schema is a component of the Redfish API standard used for providing detailed information about the firmware components, including their types and versions, within a computer system. It allows for easy management and monitoring of firmware-related aspects in a standardized manner.

- Get the firmware inventory collection

> After the BMC boots, it may take a few seconds (6-8 in NVIDIA® BlueField®-2, and 2 in BlueField-3) until everything can be seen in the following list.

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

Example output:


```
{
    "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory",
    "@odata.type": "#SoftwareInventoryCollection.SoftwareInventoryCollection",
    "Members": [
        {
            "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/BMC_Firmware"
        },
        {
            "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/Bluefield_FW_ERoT"
        },
        {
            "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_ATF"
        },
        {
            "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_ATF_pending"
        },
        {
            "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_BOARD"
        },
        {
            "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_BSP"
        },
        {
            "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_NIC"
        },
        {
            "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_NIC_pending"
        },
        {
            "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_NODE"
        },
        {
            "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_OFED"
        },
        {
            "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_OS"
        },
        {
            "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_SYS_IMAGE"
        },
        {
            "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_UEFI"
        },
        {
            "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_UEFI_pending"
        },
        {
            "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/golden_image_arm"
        },
        {
            "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/golden_image_config"
        },
        {
            "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/golden_image_nic"
        }
    ],
    "Members@odata.count": 17,
    "Name": "Software Inventory Collection"
}
```


> - Retrieving DPU object versions is supported when operating in DPU mode only.
> - In NIC mode on BF3, the supported versions are:
>   - DPU_ATF
>   - DPU_NIC
>   - DPU_UEFI
> - Pending versions are supported only on BF3

- Get a specific component information

In the following example, the `DPU_OS` inventory components are retrieved.

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

Example output:


```
{
  "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_OS",
  "@odata.type": "#SoftwareInventory.v1_4_0.SoftwareInventory",
  "Description": "Host image",
  "Id": "DPU_OS",
  "Members@odata.count": 1,
  "Name": "Software Inventory",
  "RelatedItem": [
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/Bios"
    }
  ],
  "SoftwareId": "",
  "Status": {
    "Conditions": [],
    "Health": "OK",
    "HealthRollup": "OK",
    "State": "Enabled"
  },
  "Updateable": true,
  "Version": "DOCA_2.2.0_BSP_4.2.1_Ubuntu_22.04-8.23-07"
}
```


7.5.2 Deploying Software Using BFB

NVIDIA® BlueField® devices support software deployment and upgrade through various BFB image types. For details on available image formats and their contents, refer to the "Types and Methods of Updating BlueField Software Image" page.

7.5.2.1 BFB Installation

BlueField software and firmware can be deployed using one of two methods:

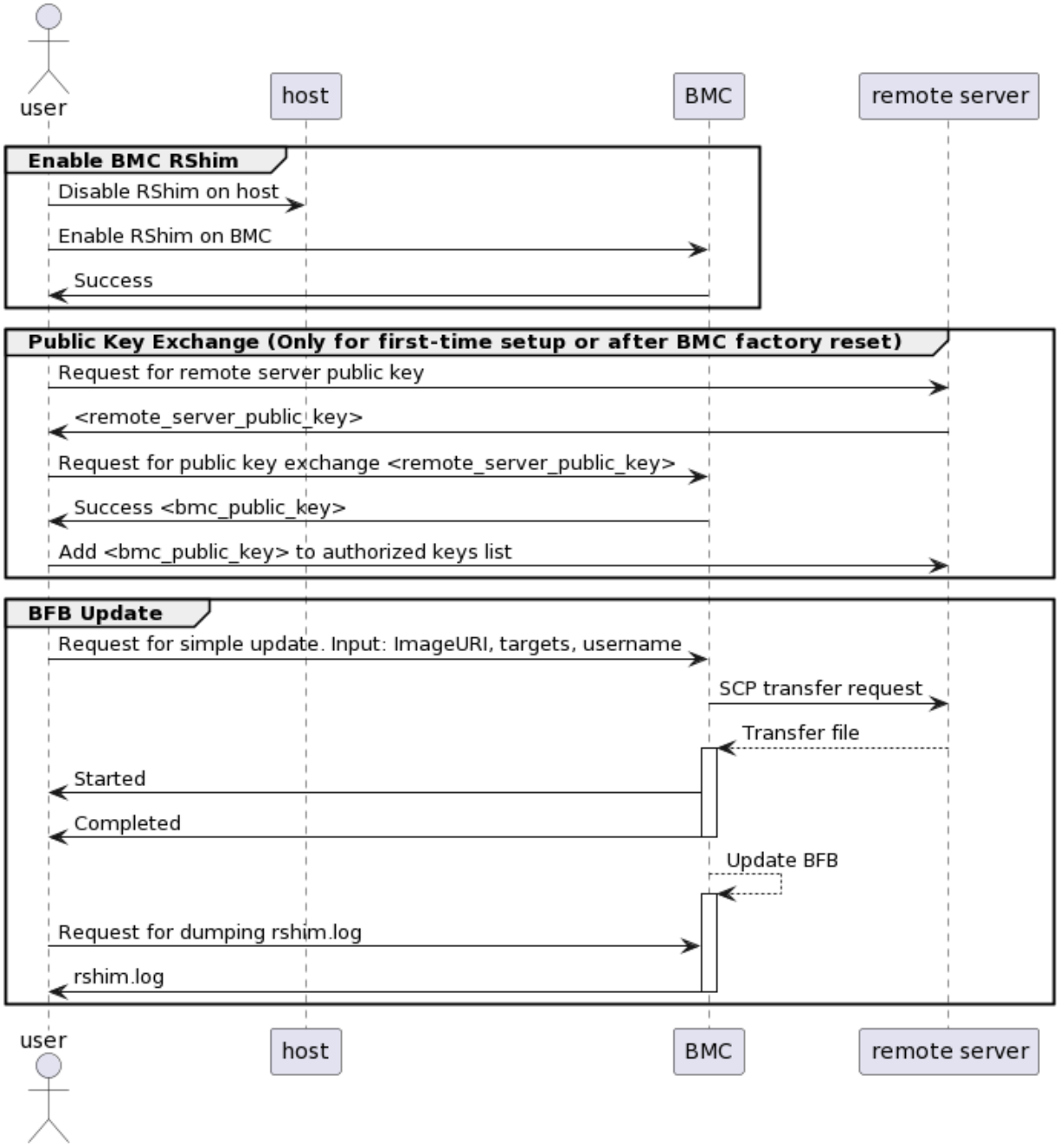
Update Flow	Description	Supported Image Types
Offline Update Flow	Traditional method where the DPU or SuperNIC is taken out of service immediately when the update begins. The device reboots into maintenance mode, applies firmware, system image, and DOCA component updates, and reboots again to activate the new versions. Ensures a clean, immediate transition but involves downtime. This flow supports recovery as well.	• BF-Bundle BFB (firmware + Arm OS + DOCA) • BF-fwBundle BFB (firmware only) • Per-SKU BF-fwBundle BFB
Deferred Update Flow	BlueField-3 supports a Deferred Update Flow, which enables administrators to update firmware and DOCA components without immediate service interruption. This capability allows a DPU or SuperNIC to continue servicing workloads while a new firmware bundle and user-space/kernel DOCA components are staged in the background. The new versions become active only after an activation command and reset are applied, minimizing downtime in production environments.	• Per-SKU BF fwBundle BFB in flat format. Use `bfb-tool` with the `--output-format flat` option to convert to flat format.

7.5.2.1.1 BFB Installation Procedure

The BFB deployment process consists of these main stages:

	Stage	Description
1	Disable RShim (if applicable)	Ensure that the RShim interface is disabled on the host side where the given DPU resides to prevent interference with the BFB update process.
2	Transfer the BFB image	Initiate the image transfer using one of the supported methods: • Redfish interface via SCP, HTTP, or HTTPS • When using SCP for the first time (or after a BMC factory reset), confirm the SSH identity of the BMC when connecting to the server hosting the BFB. • Send the update request via the Redfish API. • Direct SCP — Transfer the BFB directly to the BMC file system using secure copy.
3	Monitor installation progress	Track the update process and verify installation status through Redfish logs, BMC console, or CLI output.
4	Apply the new version	Reboot the system to activate the new firmware and software. The specific reboot behavior depends on the selected update flow (offline or deferred).

7.5.2.1.2 Update Flow



7.5.2.1.3 Changing Default Credentials Using bf.cfg

If installing the BF-Bundle BFB with BlueField Arm OS, 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`).

Alternatively, Ubuntu users can provide a unique password that will be applied at the end of the BFB installation. This password must 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.

7.5.2.1.4 Update Flow Image Transfer

7.5.2.1.4.1 Offline Update Flow

Perform the following steps to update the device using the offline flow:

1. (SuperNIC only) Unbind the PFs from the host driver before starting the update. Run the following on the host:

```
echo 0000:21:00.0 > /sys/bus/pci/drivers/mlx5_core/unbind # Example PCIe ID for the first PF
echo 0000:21:00.1 > /sys/bus/pci/drivers/mlx5_core/unbind # Example PCIe ID for the second PF (if present)
```

2. Run the following Redfish command to initiate the update:

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

> **Warning**
>
> This command initiates a soft reset on BlueField and pushes the boot stream. For NVIDIA-supplied BFBs, the eMMC is flashed automatically once the boot stream is pushed. Upon success, a `running` message is received.

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

3. (SuperNIC only) Once the installation is complete, bind the SuperNIC PFs back to the host driver:

```
echo 0000:21:00.0 > /sys/bus/pci/drivers/mlx5_core/bind # Example PCI ID for the first PF
echo 0000:21:00.1 > /sys/bus/pci/drivers/mlx5_core/bind # Example PCI ID for the second PF (if present)
```

7.5.2.1.4.2 Deferred Update Flow

> ⚠ Supported at beta level.

Deferred update flow enables upgrading DOCA components on NVIDIA® BlueField® platforms running in DPU mode while keeping services operational throughout the process. The update is applied only after a coordinated reset, minimizing downtime.

Deferred Update Flow Prerequisites

1. Download the per-SKU `fw-bundle` BFB from DOCA Downloads page.

   > ⚠ The installed firmware must be BSP 4.13.0 (DOCA 3.2.0) or later.

2. Repackage the `bf-fwbundle` in flat format using the `bfb-tool`:

   ```
   bfb-tool repack --bfb bf-fwbundle-<version>.bfb --psid <PSID> --output-format flat
   ```

   Expected output:

   ```
   BFB: .../bf-fwbundle-*.flat.bfb
   ```

3. (Optional) If a `bf.cfg` file is required, bundle it into the flat BFB using `mlx-mkbfb`.

   ```
   mlx-mkbfb --boot-args bf.cfg <input_flat.bfb> <output_cfg_flat.bfb>
   ```

4. If operating in DPU mode, add the DPU-BMC credentials to `/etc/bf-upgrade.conf` on the Arm OS using the standard `bf.cfg` format. For more details, refer to "Customizing BlueField Software Deployment".
5. (Optional) To coordinate the BlueField reboot with the host reboot, run the following on the BlueField Arm OS:

   ```
   mlxconfig -d /dev/mst/<device> set INT_CPU_AUTO_SHUTDOWN=1
   ```

   > ⚠ This must be configured in advance, as it requires a BlueField system-level reset to take effect.

Initiate Firmware Deferred Update Flow Transfer

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

> ⚠ The parameter `Stage` is only supported when `Targets` is set to `redfish/v1/UpdateService/ FirmwareInventory/DPU_OS`. Another deferred update will fail if the staging has not completed.

Example success message if the request is valid and a task is created:

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

7.5.2.1.4.3 Transfer Command Parameters

- `image_uri` - specifies the complete location of the BFB file on the remote server. It is constructed by joining the Remote Server IP and the Absolute File Path with a single forward slash (i.e., `<Remote_IP>/<Absolute_Path>`).

  > ⚠ Because absolute paths start with a slash (e.g., `/tmp/...`), the final string will typically contain a double slash (`//`) between the IP and the directory. For example, if the IP is `10.10.10.10` and the file path is `/tmp/image.bfb`, the resulting URI is: `"ImageURI": "10.10.10.10//tmp/image.bfb"`.

- `TransferProtocol` - set to either `SCP`, `HTTP`, `HTTPS`

  > ⚠ If using HTTPS, make sure the BMC has a certificate to authenticate the HTTPS server. Or install a valid certificate to authenticate:
  >
  > ```
  > curl -c cjar -b cjar -k -u root:'<password>' -X POST https://$bmc/redfish/v1/Managers/
  > Bluefield_BMC/ Truststore/Certificates -d @CAcert.json
  > ```

- `username` - username on the remote server (only required for SCP)
- `bmc_ip` - BMC IP address
- `Stage` - a value of `True` indicates a deferred flow, a value of `False` or omitting this parameter indicates an offline update flow

7.5.2.1.4.4 Setting Up Secure Connection

> ⚠ Relevant only for SCP users with Redfish.

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

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

   > ℹ OpenSSH is required for this step.

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

   Where:

- `key_type` - the type of key associated with the server storing the BFB file (e.g., ed25519)
- `remote_server_ip` - the IP address of the server hosting the BFB file

2. Retrieve the remote server's public key from the response, and send the following Redfish command to the BlueField BMC:

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

Where:

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

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


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


7.5.2.1.5 Tracking Image Transfer Status and Progress

After receiving a success message of a valid `SimpleUpdate` request and a `running` task state. Run the following Redfish command to track image transfer status and progress:

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

During the transfer, the `PercentComplete` value remains at 0 for offline update flow. If no errors occur, the `TaskState` is set to `Running`, and a keep-alive message is generated every 5 minutes. 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.

Example:


```
    {
      "@odata.type": "#MessageRegistry.v1_4_1.MessageRegistry",
      "Message": "Image 'new.bfb' is being transferred to '/dev/rshim0/boot'.",
      "MessageArgs": [
        "new.bfb",
        "/dev/rshim0/boot"
      ],
      "MessageId": "Update.1.0.TransferringToComponent",
      "Resolution": "Transfer started",
      "Severity": "OK"
    },
  …

  "PercentComplete": 60,
  "StartTime": "2024-06-10T19:39:03+00:00",
  "TaskMonitor": "/redfish/v1/TaskService/Tasks/1/Monitor",
  "TaskState": "Running",
  "TaskStatus": "OK"
```


7.5.2.1.6 Installation Status and Activation

7.5.2.1.6.1 Tracking Offline Update Flow Installation Status

In the Offline Update Flow, once the image transfer finishes, users can use the RShim miscellaneous messages log dump to track the installation's progress and status.

1. Initiate request for dump download:

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

   Where:

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

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

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

   Where:

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

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

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

   Where:

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

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

```
tar xvfJ log_dump.tar.xz
```

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

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

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

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

   Example response:

   ```
   "@odata.id": "/redfish/v1/UpdateService/FirmwareInventory/DPU_OS",
    "@odata.type": "#SoftwareInventory.v1_4_0.SoftwareInventory",
    "Description": "Host image",
    "Id": "DPU_OS",
    "Members@odata.count": 1,
    "Name": "Software Inventory",
    "RelatedItem": [
      {
        "@odata.id": "/redfish/v1/Systems/Bluefield/Bios"
      }
    ],
    "SoftwareId": "",
    "Status": {
      "Conditions": [],
      "Health": "OK",
      "HealthRollup": "OK",
      "State": "Enabled"
    },
    "Updateable": true,
    "Version": "DOCA_2.2.0_BSP_4.2.1_Ubuntu_22.04-8.23-07"
   ```

7.5.2.1.6.2 Deferred Update Flow

Checking Staging Status

Check the staging status after the transfer (i.e., the `SimpleUpdate` task) is completed successfully. A successful result of the staging procedure will be `com.nvidia.BF.Rshim.Status.Completed` after staging completes.

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X GET https://<bmc_ip>/redfish/v1/Managers/
Bluefield_BMC/Actions/Oem/NvidiaManager.GetUpdateStatus
{
  "UpdateStatus": "com.nvidia.BF.Rshim.Status.Completed"
}
```

> ⚠ The `UpdateStatus` can be:
>
> - `'com.nvidia.BF.Rshim.Status.Invalid'`
>
>   > ⚠ This references not having RShim on BMC side.
>
> - `'com.nvidia.BF.Rshim.Status.Idle'`
> - `'com.nvidia.BF.Rshim.Status.InProgress'`
> - `'com.nvidia.BF.Rshim.Status.Completed'`

- `'com.nvidia.BF.Rshim.Status.Failed'`

The default status is `com.nvidia.BF.Rshim.Status.Idle` and it take a while to update the status from `com.nvidia.BF.Rshim.Status.Idle` to `com.nvidia.BF.Rshim.Status.InProgress` after the SimpleUpdate command is sent. The final status should be `com.nvidia.BF.Rshim.Status.Completed` or `com.nvidia.BF.Rshim.Status.Failed`.

Activate the Firmware Components

Once staging is completed successfully, issue the `Activate` command. Activation is required to apply the new staged components:

```
curl -k -u root:'<password>' \
     -H "Content-Type: application/json" \
     -X POST https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Actions/Oem/NvidiaManager.Activate
```

Notes on BMC firmware activation:

- Regular BFB bundle - BMC firmware is updated without needing this manual activation command.
- PLDM BFB bundle - This activation command is required to apply the new BMC firmware.

DOCA Components Update

To complete an update to a new GA release, the DOCA Components on the Arm OS are to be updated as well. User may SSH into the DPU Arm OS and use standard Linux tools to update the DOCA components. See section "Upgrading BlueField Using Standard Linux Tools" in DOCA documentation for more details.

7.5.2.1.7 Applying New BFB Image

The following are different options for applying the new version:

Reset Type	Mode of Operation	Applying Reset Steps	Notes
Cold Boot (AC/DC Power Cycle)	• DPU Mode • NIC Mode	1. (DPU Mode only) Gracefully shut down the BlueField Arm OS. 2. Perform a full server power cycle.	• The firmware update is applied automatically during power-up. • DPU Mode only: The BlueField Arm OS must be manually shut down before the reboot; otherwise, the update will not apply.
Standard Warm Reboot	• DPU Mode • NIC Mode	1. (DPU Mode only) Gracefully shut down the BlueField Arm OS. 2. Perform a server warm reboot.	• Updates firmware and software after reboot. • DPU Mode only: The BlueField Arm OS must be manually shut down before the reboot; otherwise, the update will not apply.

Coordinated Reset (Server + DPU)	DPU Mode	1. When the administrator has completed all update flows, the DPU must be armed for the coordinated reset. Run the following command from the BlueField Arm OS. This sets a firmware trigger (`MFRL[reset_trigger]=0x48`) that instructs the DPU and its DPU-BMC to automatically reset in sync with the next host server reboot. This coordinated reset is required to apply the new firmware and software versions. `mlxreg -d /dev/mst/<device> -y --set "reset_trigger=c" --reg_name="MFRL"` 2. Perform a server warm reboot.	• Relevant to Deferred Update Flow only. • The next warm reboot will: • Gracefully shut down BlueField Arm cores • Reset the NIC, Arm Complex, and BMC • Boot from the new firmware image

7.5.2.1.8 Verify New Components are Running

After DPU reboots, check that the components have been updated:

```
curl -k -u root:'<Password>' -X GET https://<bmc ip>/redfish/v1/UpdateService/FirmwareInventory/DPU_NIC
curl -k -u root:'<Password>' -X GET https://<bmc ip>/redfish/v1/UpdateService/FirmwareInventory/DPU_ATF
curl -k -u root:'<Password>' -X GET https://<bmc ip>/redfish/v1/UpdateService/FirmwareInventory/DPU_UEFI
```

7.5.2.2 Troubleshooting Scenarios

- 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 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"
```

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

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

Where:

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

Then repeat the following steps:

a. Preparing Secure Access for Image Transfer.

b. Offline Update Flow.

- 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"
```

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"
```

7.5.3 Boot Configuration

BMC supports boot option selection commands using the Redfish or IPMI interfaces. UEFI on NVIDIA® BlueField® can query for the boot options through an IPMI/Redfish command. The BMC IPMI command supports changing the boot device selector flag only through the following options: PXE boot, or the default boot device as selected in the boot menu on BlueField. In contrast, the Redfish interface supports all available boot options.

7.5.3.1 Boot Config Using Redfish

7.5.3.1.1 Retrieving Active Boot Configuration Values

- To retrieve the active boot configuration, run:

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

> The relevant configurations would be under `Boot`.

- To retrieve all boot options (active and pending):

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

- To retrieve detailed information on a specific boot option:

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

7.5.3.1.2 Retrieving Information on Pending Boot Configurations

- To retrieve the pending boot settings:

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

- The following command retrieves only `BootOptions` configurations with a pending value different than the active one.

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

- To retrieve the details of a specific pending boot option:

```
curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/Systems/Bluefield/Settings/BootOptions/
<boot-id>
```

7.5.3.1.3 Applying Pending Boot Configurations

> Power reset of the BlueField is necessary for these changes to take effect.

- To alter the boot configuration, applying patches to the setting attribute is required:

```
curl -k -u root:'<password>' -X PATCH https://<bmc_ip>/redfish/v1/Systems/Bluefield/Settings -d '{"Boot":
{ ... }}'
```

- To set the pending boot order. The list must contain all the Boot option, even if the boot option is disabled.

```
curl -k -u root:'<password>' -X PATCH https://<bmc_ip>/redfish/v1/Systems/Bluefield/Settings/ -d
'{"Boot":{ "BootOrder": ["Boot0002",...,"BootXXX"] }}'
```

- To alter the bootOption value, currently supporting only BootOptionEnable

```
curl -k -u root:'<password>' -X PATCH https://<bmc_ip>/redfish/v1/Systems/Bluefield/Settings/BootOptions/
<Boot id> -d '{"BootOptionEnabled": false}'
```

7.5.3.1.4 Changing BootOrder Configuration

To set boot order using boot order schema, follow this procedure:

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

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

2. To get the details of a particular entity in the `BootOrder` array, perform a GET to the respective BootOption URL over 1GbE OOB to the BlueField BMC. For example, to get details of `Boot0006`, run:

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

{
    "@odata.type": "#BootOption.v1_0_3.BootOption",
    "@odata.id": "/redfish/v1/Systems/SystemId/BootOptions/Boot0006",
    "Id": "Boot0006",
    "BootOptionEnabled": true,
    "BootOptionReference": "Boot0006",
    "DisplayName": "UEFI HTTPv6 (MAC:B8CEF6B8A006)",
    "UefiDevicePath":   "PciRoot(0x0)/Pci(0x0,0x0)/Pci(0x0,0x0)/Pci(0x0,0x0)/Pci(0x0,0x0)/
MAC(B8CEF6B8A006,0x1)/
IPv6(0000:0000:0000:0000:0000:0000:0000:0000,0x0,Static,0000:0000:0000:0000:0000:0000:0000:0000,0x40,0000:0
000:0000:0000:0000:0000:0000:0000)/Uri()"
}
```

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

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

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

4. After a successful PATCH, reboot the BlueField and check if the settings have been applied by doing a GET on the `ComputerSystem` schema.
5. If the `BootOrder` array is updated as intended then the settings have been applied and the BlueField should boot as per the order in preceding cycles.
6. If `BootSourceOverrideEnabled` is set to `Once`, boot override is disabled and any related properties are reset to their former values to avoid repetition. If it is set to `Continuous`, then on every reboot, BlueField would keep performing boot override (`HTTPBoot`).

7.5.3.1.4.1 Example of Changing BootOrder Configuration

To get the supported boot options:

```
curl -k -u root:<password>' -X GET https://<bmc_ip>/redfish/v1/Systems/Bluefield/BootOptions
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions",
  "@odata.type": "#BootOptionCollection.BootOptionCollection",
  "Members": [
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0000"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot000A"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot000B"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot000C"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot000D"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot000E"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot000F"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0001"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0002"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0003"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0004"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0005"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0006"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0007"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0008"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0009"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0010"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0011"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0012"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0013"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0014"
    },
    {
```

```
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0015"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0016"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0017"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/BootOptions/Boot0040"
    }
  ],
  "Members@odata.count": 25,
  "Name": "Boot Option Collection"
}
```

To set the pending boot order settings:

> In this example, 25 boot options are present. Therefore, the command to establish the boot option order must encompass all 25 options in the active `BootOrder` list according to the desired sequence.

```
curl -k -u root:'<password>' -X PATCH https://<bmc_ip>/redfish/v1/Systems/Bluefield/Settings -d '{"Boot":
{ "BootOrder": ["Boot0040", "Boot0017", "Boot0000", "Boot0001", "Boot0002", "Boot0003", "Boot0004", "Boot0005",
"Boot0006", "Boot0007", "Boot0008", "Boot0009", "Boot000A", "Boot000B", "Boot000C", "Boot000D", "Boot000E",
"Boot000F", "Boot0010", "Boot0011", "Boot0012", "Boot0013", "Boot0014", "Boot0015", "Boot0016"] }}'
```

7.5.3.2 Boot Source Override

Boot Source Override allows administrators to remotely control the system's boot sequence without requiring physical access to configure boot order settings. This capability supports one-time or persistent boot source overrides, making it useful for automated OS deployment, system recovery, remote diagnostics, and enforcing specific security boot policies.

By dynamically setting the boot target (e.g., PXE, UEFI HTTP), administrators gain flexibility for provisioning, firmware updates, and disaster recovery workflows.

7.5.3.2.1 Boot Source Override Config Using Redfish

7.5.3.2.1.1 Get Boot Source Override Configuration

To retrieve the current boot source override configuration:

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

Example output:

```
  "Boot": {
    ...
    "BootSourceOverrideEnabled": "Disabled",
    "BootSourceOverrideMode": "UEFI",
    "BootSourceOverrideTarget": "None",
    "HttpBootUri": "path",
    "StopBootOnFault": "Never",
    "UefiTargetBootSourceOverride": "None"
    ...
  }
```

7.5.3.2.1.2 Set Boot Source Override Configuration

To set the boot source override, use:

```
curl -k -u root:'<password>' -X PATCH \
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Settings \
-d '{
  "Boot":{
    "BootSourceOverrideEnabled": "Once",
    "BootSourceOverrideMode": "UEFI",
    "BootSourceOverrideTarget": "UefiHttp",
    "UefiTargetBootSourceOverride": "None",
    "BootNext": "",
    "AutomaticRetryConfig": "Disabled"
  }
}'
```

The boot override setting take effect on the next boot and are reflected in the `redfish/v1/Systems/Bluefield` boot schema.

The following parameters can be set when configuring the boot source via the Redfish command:

- `BootSourceOverrideEnabled` -
  - `Disabled` - Disable override
  - `Once` - Apply override for next boot only
  - `Continuous` - Always use the boot override setting
- `BootSourceOverrideMode` - Must be set to `UEFI`
- `BootSourceOverrideTarget` - Must be one of the values allowed under `Boot/BootSourceOverrideTarget@Redfish.AllowableValues`
- `UefiTargetBootSourceOverride` - equired if `BootSourceOverrideTarget` is set to `UefiTarget`. Set to the `UefiDevicePath` attribute of the desired boot option.
- `BootNext` - Used if `BootSourceOverrideTarget` is set to `UefiBootnext`
- `AutomaticRetryConfig` - Only `Disabled` is supported

7.5.3.2.1.3 Examples

Set Next Boot to a Specific UEFI HTTP Target

1. Query the `BootOptions` attributes:

   ```
   curl -k -u root:'<password>' -X GET \
   https://<bmc_ip>/redfish/v1/Systems/Bluefield/BootOptions/Boot0002
   ```

   Example output:

   ```
   {
     "BootOptionReference": "Boot0002",
     "DisplayName": "NET-OOB-IPV4-HTTP",
     "UefiDevicePath": "MAC(B83FD2CA4B27,0x1)/IPv4(0.0.0.0,0x0,DHCP,...)/Uri()"
   }
   ```

2. Use the `UefiDevicePath` value as the `UefiTargetBootSourceOverride`:

   ```
   curl -k -u root:'<password>' -X PATCH \
   https://<bmc_ip>/redfish/v1/Systems/Bluefield/Settings \
   -d '{
     "Boot":{
       "BootSourceOverrideEnabled": "Once",
       "BootSourceOverrideMode": "UEFI",
       "BootSourceOverrideTarget": "UefiTarget",
       "UefiTargetBootSourceOverride": "MAC(B83FD2CA4B27,0x1)/IPv4(0.0.0.0,...)/Uri()",
       "AutomaticRetryConfig": "Disabled"
     }
   }'
   ```

Set Next Boot to PXE (Non-persistent)

```
curl -k -u root:'<password>' -X PATCH \
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Settings \
-d '{
  "Boot":{
    "BootSourceOverrideEnabled": "Once",
    "BootSourceOverrideMode": "UEFI",
    "BootSourceOverrideTarget": "Pxe"
  }
}'
```

7.5.3.2.2 Boot Source Override Config Using IPMI

The `ipmitool` utility allows configuring the Boot Source Override option, enabling the system to boot from a PXE server or another specified device.

7.5.3.2.2.1 Retrieving the Current Boot Override Settings

To view the current boot override configuration, run:

```
ipmitool chassis bootparam get 5
```

This command returns information about:

- Whether the boot override option is valid.
- Whether it is persistent or applies to the next boot only.
- The configured boot device type.

7.5.3.2.2.2 Configuring One-Time PXE Boot with Timeout

To configure a one-time PXE boot with a 60-second timeout, use the following command:

```
ipmitool chassis bootparam set bootflag force_pxe options=timeout
```

⚠ If the DPU is not reset within 60 seconds, the boot parameters will be invalidated.

7.5.3.2.2.3 Configuring One-Time PXE Boot without Timeout

To configure a one-time PXE boot without the 60-second timeout, run:

```
ipmitool chassis bootparam set bootflag force_pxe options=no-timeout
```

⚠ The boot override timer is only applicable for BlueField-3.

⚠ It is not recommended to use `ipmitool chassis bootparam` without explicitly specifying the `options` parameter, as this may result in unpredictable timer behavior.

7.5.3.2.2.4 Resetting Boot Override to Default

To clear the boot override and return to the default boot device, use:

```
ipmitool chassis bootparam set bootflag none
```

7.5.3.2.2.5 Setting Persistent PXE Boot

To configure the system to always boot from PXE (persistent override), execute:

```
ipmitool chassis bootdev pxe options=persistent
```

> The persistent option prevents the 60-second timeout from being triggered.

> If you modify bootdev or bootparam settings without explicitly specifying `options=persistent`, the persistent configuration will be disabled.

7.5.3.2.2.6 Behavior of Boot Source Override on BlueField

The Boot Source Override configuration set through the BMC remains persistent until one of the following occurs:

- It is explicitly reset to `none`.
- The BFB image is updated, which will clear the override settings.

7.6 Monitoring

This section contains the following pages:

7.6.1 System FRU

7.6.1.1 FRU Reading Redfish Commands

FRU data is embedded within the chassis schema. To retrieve the relevant FRU data, execute the following Redfish command:

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

The FRU data can be found in the following read attributes:

```
...
  "Manufacturer": "https://www.mellanox.com",
  "Model": "BlueField-3 DPU",
...
  "PartNumber": "900-9D3D4-00EN-HAR",
...
  "SerialNumber": "MT2421XZ0HDU",
...
```

7.6.1.2 FRU Reading IPMI Commands

To retrieve FRU info, run:

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

System FRU ID 0 contains information for the NVIDIA® BlueField® device:

> ⚠ The following is supported when operating in DPU mode only.

```
 FRU Device Description : Builtin FRU Device (ID 0)
 Chassis Type          : Main Server Chassis
 Chassis Part Number   : 900-9D3D4-00EN-HAA
 Chassis Serial        : N/A
 Chassis Extra         : N/A
 Chassis Extra         : ..
 Chassis Extra         : https://www.mellanox.com
 Board Mfg Date        : Mon Aug  5 16:39:00 2024
 Board Mfg             : https://www.mellanox.com
 Board Product         : BlueField-3 DPU
 Board Serial          : N/A
 Board Part Number     : 900-9D3D4-00EN-HAA
 Board Extra           : ..
 Product Manufacturer  : https://www.mellanox.com
 Product Name          : BlueField-3 DPU
 Product Part Number   : 900-9D3D4-00EN-HAA
 Product Version       : N/A
 Product Serial        : MT2430XZ0A14
 Product Asset Tag     : N/A
 Product Extra         : ..

FRU Device Description : Nvidia-BMCMezz (ID 169)
 Board Mfg Date        : Mon Aug  5 16:39:00 2024
 Board Mfg             : Nvidia
 Board Product         : Nvidia-BMCMezz
 Board Serial          : MT2430XZ0A14
 Board Part Number     : 900-9D3D4-00EN-HAA
```

To print a specific FRU:

```
ipmitool -C 17 -I lanplus -H <bmc_ip> -U ADMIN -P ADMIN fru print <fru_id>
```

To dump the binary FRU data into a file:

```
ipmitool -C 17 -I lanplus -H <bmc_ip> -U ADMIN -P ADMIN fru read <fru_id> <filename>
```

The parameter `<filename>` is the absolute path to the file.

Using the `ipmitool fru` command displays all the FRU devices detected by the BMC.

7.6.2 System Logs

7.6.2.1 System Event Logs

The System Event Log (SEL) and Event Log in OpenBMC provide robust mechanisms for monitoring, diagnosing, and troubleshooting hardware and system issues.

- SEL
  - Functionality - The SEL captures and records significant system events related to hardware and firmware. This includes events such as hardware failures, temperature thresholds, power anomalies, and other critical system changes.
  - Access - The SEL can be accessed via IPMI\Redfish commands, allowing administrators to query and retrieve logs for analysis
  - Management - Administrators can clear, save, and manage SEL entries to maintain system health and ensure critical events are recorded accurately
- Event Log:
  - Functionality - The Event Log provides a comprehensive record of both hardware and software events, offering detailed insights into system operations and potential issues. This includes firmware updates, configuration changes, security alerts, etc.
  - Access - The Event Log is accessible via Redfish interface, enabling easy retrieval and management of event data
  - Management - Users can filter, sort, and analyze events to identify patterns, diagnose problems, and improve system reliability. The Event Log supports exporting logs for offline analysis and archiving.
- Key features
  - Scalability - Both the SEL and Event Log are designed to handle a high volume of events, ensuring no critical information is lost
  - Integration - These logs integrate seamlessly with existing management tools, providing a unified view of system health and events
  - Usability - User-friendly interfaces and command-line tools make it easy to access and manage logs, ensuring administrators can quickly respond to issues

Overall, the SEL and Event Log in OpenBMC are essential tools for maintaining system integrity, improving reliability, and ensuring swift resolution of any issues that arise.

7.6.2.1.1 Event Log Redfish Commands

7.6.2.1.1.1 Displaying Event Log Information

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

Output example:

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog",
  "@odata.type": "#LogService.v1_1_0.LogService",
  "Actions": {
    "#LogService.ClearLog": {
      "target": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Actions/LogService.ClearLog"
    }
  },
  "DateTime": "2023-09-27T14:28:50+00:00",
  "DateTimeLocalOffset": "+00:00",
  "Description": "System Event Log Service",
  "Entries": {
    "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries"
  },
  "Id": "EventLog",
  "Name": "Event Log Service",
  "Oem": {
    "Nvidia": {
      "@odata.type": "#NvidiaLogService.v1_0_0.NvidiaLogService",
      "LatestEntryID": "4",
      "LatestEntryTimeStamp": "2023-09-27T14:19:30+00:00"
    }
  },
  "OverWritePolicy": "WrapsWhenFull"
}
```

7.6.2.1.1.2 Displaying List of Events

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

Output example:

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries",
  "@odata.type": "#LogEntryCollection.LogEntryCollection",
  "Description": "Collection of System Event Log Entries",
  "Members": [
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/1",
      "@odata.type": "#LogEntry.v1_9_0.LogEntry",
      "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/1/attachment",
      "Created": "2023-09-27T14:18:39+00:00",
      "EntryType": "Event",
      "Id": "1",
      "Message": "12V_ATX sensor crossed a warning low threshold going low. Reading=6.048000 Threshold=10.400000.",
      "MessageArgs": [
        "12V_ATX",
        "6.048000",
        "10.400000"
      ],
      "MessageId": "OpenBMC.0.1.SensorThresholdWarningLowGoingLow",
      "Name": "System Event Log Entry",
      "Resolution": "",
      "Resolved": false,
      "Severity": "OK"
    }
    …
  ],
  "Members@odata.count": 1,
  "Name": "System Event Log Entries"
}
```

7.6.2.1.1.3 Clearing Event Log

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X POST https://<bmc_ip>/redfish/v1/Systems/
Bluefield/LogServices/EventLog/Actions/LogService.ClearLog
```

7.6.2.1.2 SEL Redfish Commands

7.6.2.1.2.1 Displaying SEL Information

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

Output example:

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/SEL",
  "@odata.type": "#LogService.v1_1_0.LogService",
  "Actions": {
    "#LogService.ClearLog": {
      "target": "/redfish/v1/Systems/Bluefield/LogServices/SEL/Actions/LogService.ClearLog"
    }
  },
  "DateTime": "2024-07-18T10:54:52+00:00",
  "DateTimeLocalOffset": "+00:00",
  "Description": "IPMI SEL Service",
  "Entries": {
    "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/SEL/Entries"
  },
  "Id": "SEL",
  "Name": "SEL Log Service",
  "OverWritePolicy": "WrapsWhenFull"
 }
```

7.6.2.1.2.2 Displaying List of Events

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

Output example:

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/SEL/Entries",
  "@odata.type": "#LogEntryCollection.LogEntryCollection",
  "Description": "Collection of System Event Log Entries",
  "Members": [
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/SEL/Entries/1",
      "@odata.type": "#LogEntry.v1_13_0.LogEntry",
      "Created": "2024-07-16T15:34:32+00:00",
      "EntryType": "SEL",
      "Id": "1",
      "Message": "12V_ATX sensor crossed a warning low threshold going low. Reading=6.048000 Threshold=10.400000.",
      "MessageArgs": [
        "12V_ATX",
        "6.048000",
        "10.400000"
      ],
      "MessageId": "OpenBMC.0.1.SensorThresholdWarningLowGoingLow",
      "Name": "System Event Log Entry",
      "Resolution": "Check the sensor or subsystem for errors.",
      "Resolved": false,
      "Severity": "OK"
    },
   …
  ],
  "Members@odata.count": 22,
  "Name": "System Event Log Entries"
}
```

7.6.2.1.2.3 Clearing Event Log

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X POST https://<bmc_ip>/redfish/v1/Systems/
Bluefield/LogServices/EventLog/Actions/LogService.ClearLog
```

> The event log clear work in background after the above command return successfully, need wait a few seconds to have all the event clear.

7.6.2.1.2.4 Configuring SEL Info Log Capacity

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X POST https://<bmc_ip>/redfish/v1/Managers/
Bluefield_BMC/Actions/Oem/Nvidia/SelCapacity -d '{"ErrorInfoCap":300 }'
```

> It takes some time to clear the extended SEL entries if the entries count larger than the new capacity after the above command return successfully.

7.6.2.1.2.5 Getting SEL Info Log Capacity

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

Example output:

```
{
"ErrorInfoCap": 300
}
```

7.6.2.1.3 SEL IPMI Commands

The following table lists the command to use to view event logs:

7.6.2.1.3.1 Displaying SEL Information

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

7.6.2.1.3.2 Displaying List of Events

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

7.6.2.1.3.3 Displaying Extended List of Events

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

7.6.2.1.3.4 Saving SEL Events to File

```
ipmitool -C 17 -I lanplus -H <bmc_ip> -U ADMIN -P ADMIN sel save <filename>
```

7.6.2.1.3.5 Clearing SEL

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

> The SEL log clear work in background after the above command return successfully, need wait a few seconds to have all the log clear.

7.6.2.1.3.6 Configuring SEL Info Log Capacity

The capacity is a 4-byte value, and the byte order is from low to high as shown in command example.

To set the capacity to 300 lines, the value should be `0x2c 0x01 0x00 0x00`:

```
ipmitool -C 17 -I lanplus -H <bmc_ip> -U ADMIN -P ADMIN raw 0x0a 0x4a <capacity[0:7]> <capacity[8:15]>
<capacity[16:23]> <capacity[24:31]>
```

> It takes some time to clear the extended SEL entries if the entries count larger than the new capacity after the above command return successfully.

7.6.2.1.3.7 Getting SEL Info Log Capacity

```
ipmitool -C 17 -I lanplus -H <bmc_ip> -U ADMIN -P ADMIN raw 0x0a 0x4b
```

7.6.2.1.4 SEL Message Types

The following subsections detail the messages which are added to the BMC SEL and the scenarios that trigger them.

7.6.2.1.4.1 UEFI Boot

Messages are added to the BMC SEL while the BlueField UEFI is booting which describe the status of the UEFI boot.

SEL messages:

- `SMBus initialization`
- `PCI resource configuration`
- `System boot initiated`

Example:

```
SEL Record ID          : 0037
 Record Type           : 02
 Timestamp             : 06:36:06 UTC 06:36:06 UTC
 Generator ID          : 0001
 EvM Revision          : 04
 Sensor Type           : System Firmware
 Sensor Number         : 06
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data            : c207ff
 Description           : PCI resource configuration
```

7.6.2.1.4.2 DPU Watchdog Sensor

A watchdog message will be added in the event of a DPU reset caused by the hardware watchdog.

Example:

```
10 | 07/17/25 | 12:43:16 UTC | Watchdog2 #0x28 | Power cycle | Asserted
```

7.6.2.1.4.3 IPMB Sensors

QSFP Sensors

Messages are added to the SEL in case of a change in the status of the QSFP cables. The messages describe the event and status of the sensor.

List of QSFP sensors:

- `P0_link` - the QSFP 0 cable status
- `P1_link` - the QSFP 1 cable status

SEL messages:

- `Config Error` - the QSFP cable is down
- `Connected` - the QSFP cable is up

Example:

```
SEL Record ID          : 003e
 Record Type           : 02
 Timestamp             : 07:08:28 UTC 07:08:28 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Cable / Interconnect
 Sensor Number         : 00
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data (RAW)      : 010f0f
 Event Interpretation  : Missing
 Description           : Config Error

Sensor ID              : p0_link (0x0)
 Entity ID             : 31.1
 Sensor Type (Discrete): Cable / Interconnect
 States Asserted       : Cable / Interconnect
                         [Config Error]
```

Temperature Sensors

Messages are added to the SEL if temperature sensors detect a value higher than the sensor thresholds. The messages include a description of the event, BlueField FRU device description, BlueField BMC device description, and the status of the sensor.

List of temperature sensors:

- `bluefield_temp` - Bluefield temperature

- `p0_temp` - QSFP 0 cable temperature
- `p1_temp` - QSFP 1 cable temperature
- `ddr_temp` - DDR temperature

SEL messages:

- `Upper Critical going high` - crossing a upper critical threshold
- `Upper Non-critical going high` - crossing a upper non-critical threshold
- `Lower Critical going low` - crossing a lower critical threshold
- `Lower Non-critical going low` - crossing a lower non-critical threshold

Example:

```
SEL Record ID          : 003c
 Record Type           : 02
 Timestamp             : 07:01:06 UTC 07:01:06 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Temperature
 Sensor Number         : 03
 Event Type            : Threshold
 Event Direction       : Assertion Event
 Event Data (RAW)      : 592802
 Trigger Reading       : 40.000degrees C
 Trigger Threshold     : 2.000degrees C
 Description           : Upper Critical going high

Sensor ID              : p0_temp (0x3)
 Entity ID             : 0.1
 Sensor Type (Threshold)  : Temperature
 Sensor Reading        : 40 (+/- 0) degrees C
 Status                : ok
 Lower Non-Recoverable : na
 Lower Critical        : -5.000
 Lower Non-Critical    : 0.000
 Upper Non-Critical    : 70.000
 Upper Critical        : 75.000
 Upper Non-Recoverable : na
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Assertion Events      :
 Event Enable          : Event Messages Disabled
 Assertions Enabled    : lnc- lcr- unc+ ucr+
 Deassertions Enabled  : lnc+ lcr+ unc- ucr-

FRU Device Description : Nvidia-BMCMezz (ID 169)
 Board Mfg Date        : Tue Jan  3 23:16:00 2023 UTC
 Board Mfg             : Nvidia
 Board Product         : Nvidia-BMCMezz
 Board Serial          : MT2251XZ02W5
 Board Part Number     : 900-9D3B6-00CV-AAA

FRU Device Description : BlueField-3 Smar (ID 250)
 Board Mfg Date        : Tue Jan  3 23:16:00 2023 UTC
 Board Mfg             : Nvidia
 Board Product         : BlueField-3 SmartNIC Main Card
 Board Serial          : MT2251XZ02W5
 Board Part Number     : 900-9D3B6-00CV-AAA
 Product Manufacturer  : Nvidia
 Product Name          : BlueField-3 SmartNIC Main Card
 Product Part Number   : 900-9D3B6-00CV-AAA
 Product Version       : A3
 Product Serial        : MT2251XZ02W5
 Product Asset Tag     : 900-9D3B6-00CV-AAA
```

Voltage Sensors

Messages are added to the SEL if a voltage sensor's reading crosses the sensor's thresholds. The messages include a description of the event, BlueField BMC device description, and the status of the sensor.

List of voltage sensors:

- `rtc_voltage` - RTC battery voltage

SEL messages:

- `Lower Critical going low` - crossing a lower critical threshold

Example:

```
SEL Record ID          : 0227
 Record Type           : 02
 Timestamp             : 02/17/25 16:01:21 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Voltage
 Sensor Number         : 1a
 Event Type            : Threshold
 Event Direction       : Assertion Event
 Event Data (RAW)      : 52004d
 Trigger Reading       : 2.000Volts
 Trigger Threshold     : 2.302Volts
 Description           : Lower Critical going low

Sensor ID              : rtc_voltage (0x1a)
 Entity ID             : 0.1
 Sensor Type (Threshold)  : Voltage
 Sensor Reading        : 3.000 (+/- 0) Volts
 Status                : ok
 Lower Non-Recoverable : na
 Lower Critical        : 2.302
 Lower Non-Critical    : na
 Upper Non-Critical    : na
 Upper Critical        : na
 Upper Non-Recoverable : na
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Assertion Events      :
 Event Enable          : Event Messages Disabled
 Assertions Enabled    : lcr- ucr+
 Deassertions Enabled  : lcr+ ucr-

FRU Device Description : Nvidia-BMCMezz (ID 169)
 Board Mfg Date        : Sun Feb 12 07:48:00 2023 UTC
 Board Mfg             : Nvidia
 Board Product         : Nvidia-BMCMezz
 Board Serial          : MT2306XZ00BU
 Board Part Number     : 900-9D3B6-00CC-AAA
 Board Area Checksum   : OK
```

Power Sensors

Messages are added to the SEL if power sensors detect a value higher/lower than the sensor thresholds. The messages include a description of the event, BlueField BMC device description, and the status of the sensor.

List of power sensors:

- `soc_power` - current power consumption of the SoC
- `power_envelope` - maximum power consumption allowed to the SoC

SEL messages:

- `Upper Non-critical going high` - crossing an upper non-critical threshold (only in `power_envelope`)
- `Lower Critical going low` - crossing a lower critical threshold (only in `soc_power`)
- `Lower Non-critical going low` - crossing a lower non-critical threshold (only in `power_envelope`)

Example:

```
SEL Record ID          : 000e
 Record Type           : 02
 Timestamp             : 02/13/25 09:09:11 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Other
 Sensor Number         : 05
 Event Type            : Threshold
 Event Direction       : Assertion Event
 Event Data (RAW)      : 520005
 Trigger Reading       : 0.000Watts
 Trigger Threshold     : 5.000Watts
 Description           : Lower Critical going low

Sensor ID              : soc_power (0x5)
 Entity ID             : 0.1
 Sensor Type (Threshold)  : Other
```

```
 Sensor Reading        : 0 (+/- 0) Watts
 Status                : Lower Critical
 Lower Non-Recoverable : na
 Lower Critical        : 5.000
 Lower Non-Critical    : na
 Upper Non-Critical    : na
 Upper Critical        : na
 Upper Non-Recoverable : na
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Assertion Events      : lcr-
 Event Enable          : Event Messages Disabled
 Assertions Enabled    : lcr- ucr+
 Deassertions Enabled  : lcr+ ucr-

FRU Device Description : Nvidia-BMCMezz (ID 169)
 Board Mfg Date        : Mon Aug  7 07:48:00 2023 UTC
 Board Mfg             : Nvidia
 Board Product         : Nvidia-BMCMezz
 Board Serial          : MT2329XZ010Z
 Board Part Number     : 900-9D3B4-00EN-EAA
 Board Area Checksum   : OK
```

7.6.2.1.4.4 Synthesized Sensors

Power Deviation Sensors

Messages are added to the SEL if power sensors detect a value higher or lower than the sensor thresholds.

List of power deviation sensors:

- `power_envelope_deviation` - Measure the deviation between the values of the `soc_power` and `power_envelope` sensors

SEL messages:

- `Upper Critical going high` - crossing a upper critical threshold
- `Upper Non-critical going high` - crossing a upper non-critical threshold

Example:

```
SEL Record ID          : 0014
 Record Type           : 02
 Timestamp             : 02/13/25 09:17:11 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Other
 Sensor Number         : 04
 Event Type            : Threshold
 Event Direction       : Assertion Event
 Event Data (RAW)      : 590f05
 Trigger Reading       : 15.000Watts
 Trigger Threshold     : 5.000Watts
 Description           : Upper Critical going high

Sensor ID              : power_envelope_d (0x4)
 Entity ID             : 0.1
 Sensor Type (Threshold)  : Other
 Sensor Reading        : 15 (+/- 0) Watts
 Status                : Upper Critical
 Lower Non-Recoverable : na
 Lower Critical        : na
 Lower Non-Critical    : na
 Upper Non-Critical    : 0.000
 Upper Critical        : 5.000
 Upper Non-Recoverable : na
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Assertion Events      : unc+ ucr+
 Event Enable          : Event Messages Disabled
 Assertions Enabled    : lnc- lcr- unc+ ucr+
 Deassertions Enabled  : lnc+ lcr+ unc- ucr-

FRU Device Description : Nvidia-BMCMezz (ID 169)
 Board Mfg Date        : Mon Aug  7 07:48:00 2023 UTC
 Board Mfg             : Nvidia
 Board Product         : Nvidia-BMCMezz
 Board Serial          : MT2329XZ010Z
 Board Part Number     : 900-9D3B4-00EN-EAA
 Board Area Checksum   : OK
```

7.6.2.1.4.5 ADC Sensors

Messages are added to the SEL if the sensor voltage crosses the sensor's thresholds. The messages include a description of the event, BlueField FRU device description, BlueField BMC device description, and the status of the sensor.

List of ADC sensors:

- `1V_BMC`
- `1_2V_BMC`
- `1_8V`
- `1_8V_BMC`
- `2_5V`
- `3_3V`
- `3_3V_RGM`
- `5V`
- `12V_ATX`
- `12V_PCIe`
- `DVDD`
- `HVDD`
- `VDD`
- `VDDQ`
- `VDD_CPU_L`
- `VDD_CPU_R`

SEL messages:

- `Upper Non-critical going high` – crossing a upper non-critical threshold
- `Lower Non-critical going low` – crossing a lower non-critical threshold

Example:

```
SEL Record ID          : 0042
 Record Type           : 02
 Timestamp             : 09:20:50 UTC 09:20:50 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Voltage
 Sensor Number         : 06
 Event Type            : Threshold
 Event Direction       : Assertion Event
 Event Data (RAW)      : 50a9ff
 Trigger Reading       : 1.200Volts
 Trigger Threshold     : 1.810Volts
 Description           : Lower Non-critical going low

Sensor ID              : 1_2V_BMC (0x6)
 Entity ID             : 0.1
 Sensor Type (Threshold)  : Voltage
 Sensor Reading        : 1.200 (+/- 0) Volts
 Status                : ok
 Lower Non-Recoverable : na
 Lower Critical        : na
 Lower Non-Critical    : 1.143
 Upper Non-Critical    : 1.257
 Upper Critical        : na
 Upper Non-Recoverable : na
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Assertion Events      :
 Event Enable          : Event Messages Disabled
 Assertions Enabled    : lnc- unc+
 Deassertions Enabled  : lnc+ unc-

FRU Device Description : Nvidia-BMCMezz (ID 169)
 Board Mfg Date        : Tue Jan  3 23:16:00 2023 UTC
```

```
 Board Mfg             : Nvidia
 Board Product         : Nvidia-BMCMezz
 Board Serial          : MT2251XZ02W5
 Board Part Number     : 900-9D3B6-00CV-AAA

FRU Device Description : BlueField-3 Smar (ID 250)
 Board Mfg Date        : Tue Jan  3 23:16:00 2023 UTC
 Board Mfg             : Nvidia
 Board Product         : BlueField-3 SmartNIC Main Card
 Board Serial          : MT2251XZ02W5
 Board Part Number     : 900-9D3B6-00CV-AAA
 Product Manufacturer  : Nvidia
 Product Name          : BlueField-3 SmartNIC Main Card
 Product Part Number   : 900-9D3B6-00CV-AAA
 Product Version       : A3
 Product Serial        : MT2251XZ02W5
 Product Asset Tag     : 900-9D3B6-00CV-AAA
```

7.6.2.1.5 System Commands

7.6.2.1.5.1 Warm Rebooting BlueField

SEL messages:

```
System boot initiated
Initiated by warm reset
```

Example:

```
SEL Record ID          : 0001
 Record Type           : 02
 Timestamp             : 01/10/24 14:25:07 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : System Boot Initiated
 Sensor Number         : 17
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data            : 020000
 Description           : Initiated by warm reset
```

7.6.2.1.5.2 Hard Rebooting BlueField

SEL messages:

```
System boot initiated
Initiated by hard reset
```

Example:

```
SEL Record ID          : 0008
 Record Type           : 02
 Timestamp             : 01/10/24 14:33:01 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : System Boot Initiated
 Sensor Number         : 17
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data            : 010000
 Description           : Initiated by hard reset
```

If the host does not assert the `PERST` / `ALL_STANDBY` signal, causing the reset to fail, the following SEL messages can be observed:

```
Power Unit
Failure detected
```

Example:

```
SEL Record ID          : 0004
 Record Type           : 02
 Timestamp             : 07/25/24 13:32:18 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Power Unit
 Sensor Number         : 1b
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data            : 060000
 Description           : Failure detected
```

7.6.2.1.5.3 Shutting Down BlueField

SEL messages:

```
OS Critical Stop
OS graceful shutdown
```

Example:

```
SEL Record ID          : 000a
 Record Type           : 02
 Timestamp             : 01/10/24 14:34:45 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : OS Critical Stop
 Sensor Number         : 18
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data            : 030000
 Description           : OS graceful shutdown
```

7.6.2.1.5.4 Updating BlueField BFB Image

SEL messages:

```
Firmware or software change success
```

Example:

```
SEL Record ID          : 0007
 Record Type           : 02
 Timestamp             : 06/11/24 14:03:02 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Version Change
 Sensor Number         : 18
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data            : c70000
 Description           : Firmware or software change success
```

7.6.2.1.5.5 Updating BMC

SEL messages:

```
Firmware or software change success, Mngmt SW agent change
```

Example:

```
SEL Record ID          : 0010
 Record Type           : 02
 Timestamp             : 01/10/24 15:48:01 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Version Change
 Sensor Number         : 19
```

```
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data            : c70e00
 Description           : Firmware or software change success, Mngmt SW agent change
```

7.6.2.1.6 RAS Errors

7.6.2.1.6.1 Multi-bit ECC

SEL messages:

```
Uncorrectable ECC
```

Example:

```
SEL Record ID          : 024a
 Record Type           : 02
 Timestamp             : 06/20/24 15:54:58 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Memory
 Sensor Number         : 17
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data            : 010000
 Description           : Uncorrectable ECC
```

7.6.2.1.6.2 Single-bit ECC

SEL messages:

```
Correctable ECC
```

Example:

```
SEL Record ID          : 0254
 Record Type           : 02
 Timestamp             : 06/20/24 16:01:05 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Memory
 Sensor Number         : 17
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data            : 000000
 Description           : Correctable ECC
```

7.6.2.1.6.3 Cache Correctable Error

- Event `data1 0x0C` indicates Correctable machine check error
- Event `data2 0x1` indicates a cache error (same Processor Error Type enumeration used by Cper)
  https://uefi.org/specs/UEFI/2.10_A/Apx_N_Common_Platform_Error_Record.html

SEL messages:

```
Correctable machine check error
```

Example:

```
SEL Record ID          : 009d
 Record Type           : 02
```

```
 Timestamp             : 12/17/24 12:16:35 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Processor
 Sensor Number         : 18
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data            : 0c0100
 Description           : Correctable machine check error
```

7.6.2.1.6.4 Cache Uncorrectable Error

- Event `data1 0x0C` indicates Correctable machine check error
- Event `data2 0x1` indicates a cache error (same Processor Error Type enumeration used by Cper) https://uefi.org/specs/UEFI/2.10_A/Apx_N_Common_Platform_Error_Record.html

SEL messages:

```
Uncorrectable machine check exception
```

Example:

```
SEL Record ID          : 0012
 Record Type           : 02
 Timestamp             : 12/10/24 16:32:27 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Processor
 Sensor Number         : 1b
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data            : 0b0100
 Description           : Uncorrectable machine check exception
```

7.6.2.1.6.5 Cache Uncorrectable Fatal Error

- Event `data1 0x0C` indicates Correctable machine check error
- Event `data2 0x1` indicates a cache error (same Processor Error Type enumeration used by Cper)
- Event `data3 0x1` indicates a fatal error. https://uefi.org/specs/UEFI/2.10_A/Apx_N_Common_Platform_Error_Record.html

SEL messages:

```
Uncorrectable machine check exception
```

Example:

```
SEL Record ID          : 00b1
 Record Type           : 02
 Timestamp             : 12/17/24 16:07:11 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Processor
 Sensor Number         : 18
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data            : 0b0101
 Description           : Uncorrectable machine check exception
```

7.6.2.1.6.6 PCIe Correctable Error

SEL messages:

```
Bus Correctable error
```

Example:

```
SEL Record ID          : 000c
 Record Type           : 02
 Timestamp             : 02/10/25 15:11:22 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Critical Interrupt
 Sensor Number         : ff
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data            : 070000
 Description           : Bus Correctable error
```

7.6.2.1.6.7 PCIe Uncorrectable Error

SEL messages:

```
Bus Uncorrectable error
```

Example:

```
SEL Record ID         : 001c
 Record Type          : 02
 Timestamp            : 02/12/25 09:30:22 UTC
 Generator ID         : 0020
 EvM Revision         : 04
 Sensor Type          : Critical Interrupt
 Sensor Number        : ff
 Event Type           : Sensor-specific Discrete
 Event Direction      : Assertion Event
 Event Data           : 080000
 Description          : Bus Uncorrectable error
```

7.6.2.1.6.8 PCIe Fatal Error

SEL messages:

```
Bus Fatal Error
```

Example:

```
SEL Record ID          : 0012
 Record Type           : 02
 Timestamp             : 02/12/25 12:10:25 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Critical Interrupt
 Sensor Number         : ff
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data            : 0a0000
 Description           : Bus Fatal Error
```

7.6.2.1.6.9 ATX Power Error

SEL messages:

```
Power Supply
Failure detected
```

Example:

```
SEL Record ID          : 0006
 Record Type           : 02
 Timestamp             : 02/17/25 13:47:28 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Power Supply
 Sensor Number         : 02
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data (RAW)      : 010000
 Event Interpretation  : Missing
 Description           : Failure detected
```

7.6.2.1.7 Arm Frequency Change

The system's frequency is dynamically managed by the Arm cores, based on the system's power consumption and temperature. As long as they stay below a predefined threshold, the Arm cores operate at full frequency. If power consumption or temperature exceeds their threshold, the frequency is reduced in stages for mitigation. This reduction will put the system under the crossed threshold, and then the frequency will be throttled back to full performance.

SEL message:

```
Throttled | Asserted
```

Example:

```
SEL Record ID          : 0004
 Record Type           : 02
 Timestamp             : 09/01/24 09:12:34 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Processor
 Sensor Number         : ff
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data            : 0a0000
 Description           : Throttled
```

More details can be extracted from Redfish. Further information is available in section "Redfish Event Log".

7.6.2.1.8 Data Port Module Events

7.6.2.1.8.1 Data Port Module High Power Consumption Notification

An SEL entry is generated when the power consumption of a data port module exceeds a critical threshold.

SEL messages:

```
Voltage <sensor-id> | Upper Non-recoverable going high | Asserted
```

Example:

```
SEL Record ID          : 0029
Record Type           : 02
Timestamp             : 09/29/24 13:22:44 UTC
Generator ID          : 0020
EvM Revision          : 04
Sensor Type           : Voltage
Sensor Number         : 1d
Event Type            : Threshold
```

```
Event Direction       : Assertion Event
Event Data            : 0b0000
Description           : Upper Non-recoverable going high
```

The sensor ID can be found using `ipmitool sdr list all -vv`.

- Port 0 sensor name: `voltage_p0`
- Port 1 sensor name: `voltage_p1`

7.6.2.1.8.2 Data Port Module Thermal "Going High" Notification

Indicates that the temperature of the data port module exceeded valid range.

SEL messages:

```
Temperature <sensor-id> | Upper Non-critical going high | Asserted
```

Example:

```
SEL Record ID          : 002c
 Record Type           : 02
 Timestamp             : 10/01/24 06:47:54 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Temperature
 Sensor Number         : 1d
 Event Type            : Threshold
 Event Direction       : Assertion Event
 Event Data            : 070000
 Description           : Upper Non-critical going high
```

The sensor ID can be found using `ipmitool sdr list all -vv`.

- Port 0 sensor name: `thermal_p0`
- Port 1 sensor name: `thermal_p1`

7.6.2.1.8.3 Data Port Module Thermal "Going Low" Notification

Indicates that the temperature of the data port module returned to valid range.

SEL messages:

```
Temperature <sensor-id> | Upper Non-critical going low  | Asserted
```

Example:

```
SEL Record ID          : 002d
 Record Type           : 02
 Timestamp             : 10/01/24 06:47:58 UTC
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Temperature
 Sensor Number         : 1d
 Event Type            : Threshold
 Event Direction       : Assertion Event
 Event Data            : 060000
 Description           : Upper Non-critical going low
```

The sensor ID can be found using `ipmitool sdr list all -vv`.

- Port 0 sensor name: `thermal_p0`
- Port 1 sensor name: `thermal_p1`

7.6.2.2 Redfish Event Log

7.6.2.2.1 System Commands

7.6.2.2.1.1 Warm Rebooting BlueField


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/2",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/2/attachment",
  "Created": "2024-07-26T10:41:57+00:00",
  "EntryType": "Event",
  "Id": "2",
  "Message": "DPU Warm Reset",
  "Modified": "2024-07-26T10:41:57+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.2 Hard Rebooting BlueField


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/7",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/7/attachment",
  "Created": "2024-07-26T09:50:16+00:00",
  "EntryType": "Event",
  "Id": "7",
  "Message": "DPU Hard Reset",
  "Modified": "2024-07-26T09:50:16+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


If the host does not assert the PERST signal, causing the reset to fail:


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/8",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/8/attachment",
  "Created": "2024-07-26T09:58:34+00:00",
  "EntryType": "Event",
  "Id": "8",
  "Message": "PERST is in de-assert, skip SoC Hard Reset",
  "Modified": "2024-07-26T09:58:34+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


If the host does not assert the `ALL_STANDBY` signal, causing the reset to fail:


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/8",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/8/attachment",
  "Created": "2024-07-26T09:58:34+00:00",
  "EntryType": "Event",
  "Id": "8",
  "Message": "ALL_STDBY is in de-assert, skip SoC Hard Reset",
  "Modified": "2024-07-26T09:58:34+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```

7.6.2.2.1.3 Shutting Down BlueField


```
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/18",
      "@odata.type": "#LogEntry.v1_13_0.LogEntry",
      "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/18/attachment",
      "Created": "2024-07-26T13:56:46+00:00",
      "EntryType": "Event",
      "Id": "18",
      "Message": "DPU Shutdown",
      "Modified": "2024-07-26T13:56:46+00:00",
      "Name": "System Event Log Entry",
      "Resolved": false,
      "Severity": "OK"
    },
```


7.6.2.2.1.4 Updating BMC


```
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/2",
      "@odata.type": "#LogEntry.v1_13_0.LogEntry",
      "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/2/attachment",
      "Created": "2024-07-26T10:41:57+00:00",
      "EntryType": "Event",
      "Id": "2",
      "Message": "BMC SW update",
      "Modified": "2024-07-26T10:41:57+00:00",
      "Name": "System Event Log Entry",
      "Resolved": false,
      "Severity": "OK"
    },
```


7.6.2.2.1.5 Getting Measurements


```
{
      "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/12",
      "@odata.type": "#LogEntry.v1_15_0.LogEntry",
      "Created": "2025-03-04T15:34:43+00:00",
      "EntryType": "Event",
      "Id": "12",
      "Message": "Redfish attestation measurements POST request received",
      "Modified": "2025-03-04T15:34:43+00:00",
      "Name": "System Event Log Entry",
      "Resolved": false,
      "Severity": "OK"
    }
```


7.6.2.2.1.6 Adding BMC User


```
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/3",
      "@odata.type": "#LogEntry.v1_13_0.LogEntry",
      "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/3/attachment",
      "Created": "2024-01-10T14:25:14+00:00",
      "EntryType": "Event",
      "Id": "3",
      "Message": "BMC User Create test0",
      "Modified": "2024-01-10T14:25:14+00:00",
      "Name": "System Event Log Entry",
      "Resolved": false,
      "Severity": "OK"
    }
```


7.6.2.2.1.7 Deleting BMC User


```
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/2",
      "@odata.type": "#LogEntry.v1_13_0.LogEntry",
      "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/2/attachment",
      "Created": "2024-01-10T14:25:14+00:00",
      "EntryType": "Event",
      "Id": "2",
      "Message": "BMC User Delete test0",
      "Modified": "2024-01-10T14:25:14+00:00",
      "Name": "System Event Log Entry",
```

```
    "Resolved": false,
    "Severity": "OK"
  }
```


7.6.2.2.1.8 Renaming BMC User


```
  {
    "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/2",
    "@odata.type": "#LogEntry.v1_13_0.LogEntry",
    "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/2/attachment",
    "Created": "2024-01-10T14:25:14+00:00",
    "EntryType": "Event",
    "Id": "2",
    "Message": "BMC User Rename test0 To test1",
    "Modified": "2024-01-10T14:25:14+00:00",
    "Name": "System Event Log Entry",
    "Resolved": false,
    "Severity": "OK"
  }
```


7.6.2.2.1.9 BMC User Login


```
  {
    "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/27",
    "@odata.type": "#LogEntry.v1_13_0.LogEntry",
    "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/27/attachment",
    "Created": "2024-06-11T13:07:34+00:00",
    "EntryType": "Event",
    "Id": "27",
    "Message": "User (root) logged in",
    "Modified": "2024-06-11T13:07:34+00:00",
    "Name": "System Event Log Entry",
    "Resolved": false,
    "Severity": "OK"
  }
```


7.6.2.2.1.10 BMC User Logout


```
  {
    "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/37",
    "@odata.type": "#LogEntry.v1_13_0.LogEntry",
    "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/37/attachment",
    "Created": "2024-06-11T13:30:48+00:00",
    "EntryType": "Event",
    "Id": "37",
    "Message": "User (root) logged out",
    "Modified": "2024-06-11T13:30:48+00:00",
    "Name": "System Event Log Entry",
    "Resolved": false,
    "Severity": "OK"
  }
```


7.6.2.2.1.11 Changing BMC User Password


```
  {
    "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/11",
    "@odata.type": "#LogEntry.v1_13_0.LogEntry",
    "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/11/attachment",
    "Created": "2024-06-11T13:03:42+00:00",
    "EntryType": "Event",
    "Id": "11",
    "Message": "Password changed for root",
    "Modified": "2024-06-11T13:03:42+00:00",
    "Name": "System Event Log Entry",
    "Resolved": false,
    "Severity": "OK"
  }
```


7.6.2.2.1.12 Changing BlueField UEFI Password


```
  {
    "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/7",
    "@odata.type": "#LogEntry.v1_13_0.LogEntry",
    "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/7/attachment",
    "Created": "2024-06-11T13:02:04+00:00",
```

```
  "EntryType": "Event",
  "Id": "7",
  "Message": "Password changed for UEFI",
  "Modified": "2024-06-11T13:02:04+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```

7.6.2.2.1.13 Adding BMC IP Address

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/20",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/20/attachment",
  "Created": "2024-07-25T13:40:22+00:00",
  "EntryType": "Event",
  "Id": "20",
  "Message": "BMC IP Address Added",
  "Modified": "2024-07-25T13:40:22+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
},
```

7.6.2.2.1.14 Deleting BMC IP Address

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/21",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/21/attachment",
  "Created": "2024-01-10T15:53:57+00:00",
  "EntryType": "Event",
  "Id": "21",
  "Message": "BMC IP Address Deleted",
  "Modified": "2024-01-10T15:53:57+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```

7.6.2.2.1.15 Changing BMC IPv4 Mode to Static

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/6",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/6/attachment",
  "Created": "2024-06-11T13:02:04+00:00",
  "EntryType": "Event",
  "Id": "6",
  "Message": "Set IPv4 to Static mode",
  "Modified": "2024-06-11T13:02:04+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
},
```

7.6.2.2.1.16 Changing BMC IPv4 Mode to DHCP

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/9",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/9/attachment",
  "Created": "2024-06-11T13:02:05+00:00",
  "EntryType": "Event",
  "Id": "9",
  "Message": "Set IPv4 to DHCP mode",
  "Modified": "2024-06-11T13:02:05+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```

7.6.2.2.1.17 Changing BMC IPv6 Mode to Static


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/38",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/38/attachment",
  "Created": "2024-06-11T13:34:57+00:00",
  "EntryType": "Event",
  "Id": "38",
  "Message": "Set IPv6 to Static mode",
  "Modified": "2024-06-11T13:34:57+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.18 Changing BMC IPv6 Mode to DHCP


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/39",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/39/attachment",
  "Created": "2024-06-11T13:35:03+00:00",
  "EntryType": "Event",
  "Id": "39",
  "Message": "Set IPv6 to DHCP mode",
  "Modified": "2024-06-11T13:35:03+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.19 Changing BMC NTP Server


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/8",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/8/attachment",
  "Created": "2024-06-11T14:07:30+00:00",
  "EntryType": "Event",
  "Id": "8",
  "Message": "BMC NTP Servers Changed",
  "Modified": "2024-06-11T14:07:30+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.20 Starting RShim on BMC


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/4",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/4/attachment",
  "Created": "2024-06-11T13:00:41+00:00",
  "EntryType": "Event",
  "Id": "4",
  "Message": "Started rshim service on BMC",
  "Modified": "2024-06-11T13:00:41+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.21 Stopping RShim on BMC


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/35",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/35/attachment",
  "Created": "2024-06-11T13:29:19+00:00",
  "EntryType": "Event",
  "Id": "35",
  "Message": "Stopped rshim service on BMC",
  "Modified": "2024-06-11T13:29:19+00:00",
```

```
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.22 Reset of TOR E-Switch


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/32",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/32/attachment",
  "Created": "2024-06-11T13:19:57+00:00",
  "EntryType": "Event",
  "Id": "32",
  "Message": "Reset of TOR E-Switch",
  "Modified": "2024-06-11T13:19:57+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.23 Setting Mode of 3-port Switch Ports to Allow All Ports to Access OOB RJ45


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/34",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/34/attachment",
  "Created": "2024-06-11T13:20:12+00:00",
  "EntryType": "Event",
  "Id": "34",
  "Message": "All ports are allowed access to RJ45",
  "Modified": "2024-06-11T13:20:12+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.24 Setting Mode of 3-port Switch Ports to Allow Only BMC Port to Access OOB RJ45


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/33",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/33/attachment",
  "Created": "2024-06-11T13:20:09+00:00",
  "EntryType": "Event",
  "Id": "33",
  "Message": "Only BMC port is allowed access to RJ45",
  "Modified": "2024-06-11T13:20:09+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.25 Clearing BMC SEL


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/2",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/2/attachment",
  "Created": "2024-06-11T13:49:03+00:00",
  "EntryType": "Event",
  "Id": "2",
  "Message": "Start clearing SEL",
  "Modified": "2024-06-11T13:49:03+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.26 BMC Factory Reset


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/1",
```

```
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/1/attachment",
  "Created": "2024-06-11T13:49:03+00:00",
  "EntryType": "Event",
  "Id": "1",
  "Message": "BMC factory reset will take effect upon reboot",
  "Modified": "2024-06-11T13:49:03+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.27 Resetting BMC Soft


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/17",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/17/attachment",
  "Created": "2024-01-10T15:52:46+00:00",
  "EntryType": "Event",
  "Id": "17",
  "Message": "BMC Soft Reset",
  "Modified": "2024-01-10T15:52:46+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.28 Enabling RShim Access from Host


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/3",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/3/attachment",
  "Created": "2024-06-11T13:51:28+00:00",
  "EntryType": "Event",
  "Id": "3",
  "Message": "RShim access privilege from host will be enabled after NIC reset",
  "Modified": "2024-06-11T13:51:28+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.29 Disabling RShim Access from Host


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/4",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/4/attachment",
  "Created": "2024-06-11T13:51:29+00:00",
  "EntryType": "Event",
  "Id": "4",
  "Message": "RShim access privilege from host will be disabled after NIC reset",
  "Modified": "2024-06-11T13:51:29+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.30 Enabling BlueField DPU Mode


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/31",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/31/attachment",
  "Created": "2024-06-11T13:18:40+00:00",
  "EntryType": "Event",
  "Id": "31",
  "Message": "DPU mode will take effect after NIC reset",
  "Modified": "2024-06-11T13:18:40+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```

7.6.2.2.1.31 Enabling BlueField NIC Mode


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/30",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/30/attachment",
  "Created": "2024-06-11T13:18:39+00:00",
  "EntryType": "Event",
  "Id": "30",
  "Message": "NIC mode will take effect after NIC reset",
  "Modified": "2024-06-11T13:18:39+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.32 Enabling BlueField Secure Boot


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/28",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/28/attachment",
  "Created": "2024-06-11T13:14:34+00:00",
  "EntryType": "Event",
  "Id": "28",
  "Message": "Secure Boot Option changed to Enable",
  "Modified": "2024-06-11T13:14:34+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.33 Disabling BlueField Secure Boot


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/29",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/29/attachment",
  "Created": "2024-06-11T13:14:45+00:00",
  "EntryType": "Event",
  "Id": "29",
  "Message": "Secure Boot Option changed to Disable",
  "Modified": "2024-06-11T13:14:45+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.34 Changing BlueField Boot Order


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/6",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/6/attachment",
  "Created": "2024-06-11T13:02:04+00:00",
  "EntryType": "Event",
  "Id": "6",
  "Message": "System boot order changed",
  "Modified": "2024-06-11T13:02:04+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.35 Enabling BlueField Boot Source


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/4",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/4/attachment",
  "Created": "2024-07-26T09:49:42+00:00",
  "EntryType": "Event",
  "Id": "4",
  "Message": "System boot source enabled",
  "Modified": "2024-07-26T09:49:42+00:00",
```

```
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```

7.6.2.2.1.36 Disabling BlueField Boot Source

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/5",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/5/attachment",
  "Created": "2024-07-26T09:49:42+00:00",
  "EntryType": "Event",
  "Id": "5",
  "Message": "System boot source disabled,
  "Modified": "2024-07-26T09:49:42+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
},
```

7.6.2.2.1.37 Changing BlueField Boot Source from Continuous to Once

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/3",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/3/attachment",
  "Created": "2024-07-26T09:49:42+00:00",
  "EntryType": "Event",
  "Id": "3",
  "Message": "System boot source will take effect for one boot",
  "Modified": "2024-07-26T09:49:42+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
},
```

This log will not be generated if only the boot source is enabled without switching the boot override persistent setting

7.6.2.2.1.38 Changing BlueField Boot Source from Once to Continuous

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/3",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/3/attachment",
  "Created": "2024-07-26T10:42:31+00:00",
  "EntryType": "Event",
  "Id": "3",
  "Message": "System boot source will take effect continuously",
  "Modified": "2024-07-26T10:42:31+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```

This log will not be generated if only the boot source is enabled without switching the boot override persistent setting

7.6.2.2.1.39 Changing BlueField Boot Source to Default

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/11",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/11/attachment",
  "Created": "2024-06-11T14:12:12+00:00",
  "EntryType": "Event",
  "Id": "11",
  "Message": "System boot source changed to Default",
```

```
    "Modified": "2024-06-11T14:12:12+00:00",
    "Name": "System Event Log Entry",
    "Resolved": false,
    "Severity": "OK"
  }
```

7.6.2.2.1.40 Changing BlueField Boot Source to PXE

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/12",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/12/attachment",
  "Created": "2024-06-11T14:12:13+00:00",
  "EntryType": "Event",
  "Id": "12",
  "Message": "System boot source changed to Network",
  "Modified": "2024-06-11T14:12:13+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```

7.6.2.2.1.41 Changing BlueField Boot Source to UEFI HTTP

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/12",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/12/attachment",
  "Created": "2024-06-11T14:12:13+00:00",
  "EntryType": "Event",
  "Id": "12",
  "Message": "System boot source changed to HTTP",
  "Modified": "2024-06-11T14:12:13+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```

7.6.2.2.1.42 Changing BlueField Boot Type to Legacy

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/9",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/9/attachment",
  "Created": "2024-06-11T14:09:40+00:00",
  "EntryType": "Event",
  "Id": "9",
  "Message": "System boot type changed to Legacy",
  "Modified": "2024-06-11T14:09:40+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```

7.6.2.2.1.43 Changing BlueField Boot Type to UEFI

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/10",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/10/attachment",
  "Created": "2024-06-11T14:10:43+00:00",
  "EntryType": "Event",
  "Id": "10",
  "Message": "System boot type changed to UEFI",
  "Modified": "2024-06-11T14:10:43+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```

7.6.2.2.1.44 Updating BlueField BFB Image

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/6",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
```

```
    "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/6/attachment",
    "Created": "2024-06-11T14:01:13+00:00",
    "EntryType": "Event",
    "Id": "6",
    "Message": "Starting Bluefield DPU BFB update",
    "Modified": "2024-06-11T14:01:13+00:00",
    "Name": "System Event Log Entry",
    "Resolved": false,
    "Severity": "OK"
}
```


7.6.2.2.1.45 Arm Frequency Change Redfish System Command

3 optional message descriptions:

- CPU frequency switched to P0 [100%].
- CPU frequency switched to P1 [80%].
- CPU frequency switched to P2 [50%].

Example:


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/5",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/5/attachment",
  "Created": "2024-09-01T09:12:46+00:00",
  "EntryType": "Event",
  "Id": "5",
  "Message": "CPU frequency switched to P0 [100%].",
  "Modified": "2024-09-01T09:12:46+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```


7.6.2.2.1.46 Data Port Module High Power Consumption Notification

An SEL entry generated when the power consumption of a data port module exceeds a critical threshold.


```
{
   "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/764",
   "@odata.type": "#LogEntry.v1_13_0.LogEntry",
   "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/764/attachment",
   "Created": "2024-09-29T08:56:54+00:00",
   "EntryType": "Event",
   "Id": "764",
   "Message": "SEL event for port 1 High Module Current notification, ThresholdCriticalHighGoingHigh",
   "Modified": "2024-09-29T08:56:54+00:00",
   "Name": "System Event Log Entry",
   "Resolved": false,
   "Severity": "Critical"
}
```


7.6.2.2.1.47 Data Port Module Temperature Going High

Indicates that data port module temperature exceeded valid range.


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/SEL/Entries/5",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "Created": "2024-09-16T07:41:13+00:00",
  "EntryCode": "Assert",
  "EntryType": "SEL",
  "Id": "5",
  "Message": "SEL event for port 0 high thermal notification, ThresholdWarningHighGoingHigh",
  "MessageId": "SEL event for port 0 high thermal notification, ThresholdWarningHighGoingHigh",
  "Modified": "2024-09-16T07:41:13+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "SensorNumber": 28,
  "SensorType": "Temperature",
  "Severity": "Warning"
}
```

7.6.2.2.1.48 Data Port Module Temperature Going Low

Indicates that data port module temperature returned to valid range.


```
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/SEL/Entries/6",
      "@odata.type": "#LogEntry.v1_13_0.LogEntry",
      "Created": "2024-09-16T07:41:19+00:00",
      "EntryCode": "Assert",
      "EntryType": "SEL",
      "Id": "6",
      "Message": "SEL event for port 0 normal thermal notification, ThresholdGoingLow",
      "MessageId": "SEL event for port 0 normal thermal notification, ThresholdGoingLow",
      "Modified": "2024-09-16T07:41:19+00:00",
      "Name": "System Event Log Entry",
      "Resolved": false,
      "SensorNumber": 28,
      "SensorType": "Temperature",
      "Severity": "OK"
    }
```


7.6.2.2.2 RAS Logging

CPER to Redfish severity translation:

CPER Severity	Redfish Severity
Recoverable	Warning
Fatal	Critical
Corrected	OK
Informational	Warning

7.6.2.2.2.1 RAS Cache Error


```
{
   "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/3",
   "@odata.type": "#LogEntry.v1_15_0.LogEntry",
   "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/3/attachment",
   "CPER": {
       "NotificationType": "09a9d5ac-5204-4214-96e5-94992e752bcd",
       "Oem": {
           "Nvidia": {
               "@odata.type": "#NvidiaCPER.v1_0_0.NvidiaCPER",
               "ArmProcessor": {
                   "ContextInfo": [],
                   "ContextInfoNum": 0,
                   "ErrorAffinity": {
                       "Type": "Vendor Defined",
                       "Value": 0
                   },
                   "ErrorInfo": [
                       {
                           "CacheError": {
                               "Corrected": false,
                               "Level": 0,
                               "Operation": {
                                   "Name": "Generic Error",
                                   "Value": 0
                               },
                               "PrecisePC": false,
                               "ProcessorContextCorrupt": false,
                               "RestartablePC": false,
                               "TransactionType": {
                                   "Name": "Instruction",
                                   "Value": 0
                               },
                               "ValidationBits": {
                                   "CorrectedValid": false,
                                   "LevelValid": false,
                                   "OperationValid": false,
                                   "PrecisePCValid": false,
                                   "ProcessorContextCorruptValid": false,
                                   "RestartablePCValid": false,
                                   "TransactionTypeValid": false
                               }
                           },
                           "ErrorType": {
                               "Name": "Cache Error",
```

```
                            "Value": 0
                        },
                        "Flags": {
                            "FirstErrorCaptured": false,
                            "LastErrorCaptured": false,
                            "Overflow": false,
                            "Propagated": false
                        },
                        "Length": 32,
                        "MultipleError": {
                            "Type": "Multiple Errors",
                            "Value": 1
                        },
                        "PhysicalFaultAddress": 0,
                        "ValidationBits": {
                            "ErrorInformationValid": false,
                            "FlagsValid": false,
                            "MultipleErrorValid": true,
                            "PhysicalFaultAddressValid": false,
                            "VirtualFaultAddressValid": false
                        },
                        "Version": 0,
                        "VirtualFaultAddress": 0
                    }
                ],
                "ErrorInfoNum": 1,
                "MidrEl1": 1091556385,
                "MpidrEl1": 2164326400,
                "PsciState": 0,
                "Running": true,
                "SectionLength": 72,
                "ValidationBits": {
                    "ErrorAffinityLevelValid": false,
                    "MpidrValid": true,
                    "RunningStateValid": true,
                    "VendorSpecificInfoValid": false
                }
            }
        }
    },
    "SectionType": "e19e3d16-bc11-11e4-9caa-c2051d5d46b0"
},
"Created": "2024-11-15T19:14:48+00:00",
"DiagnosticDataType": "CPERSection",
"EntryType": "Event",
"Id": "3",
"Message": "A platform error occurred.",
"MessageArgs": [],
"MessageId": "Platform.1.0.PlatformError",
"Name": "System Event Log Entry",
"Resolution": "Check additional diagnostic data if available.",
"Resolved": false,
"Severity": "Warning"
}
```


7.6.2.2.2.2 RAS Memory Error


```
{
    "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/6",
    "@odata.type": "#LogEntry.v1_15_0.LogEntry",
    "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/6/attachment",
    "CPER": {
        "NotificationType": "09a9d5ac-5204-4214-96e5-94992e752bcd",
        "Oem": {
            "Nvidia": {
                "@odata.type": "#NvidiaCPER.v1_0_0.NvidiaCPER",
                "Memory": {
                    "Bank": {
                        "Value": 0
                    },
                    "BitPosition": 0,
                    "Card": 0,
                    "CardSmbiosHandle": 0,
                    "Column": 0,
                    "Device": 0,
                    "ErrorStatus": {
                        "AddressSignal": true,
                        "ControlSignal": false,
                        "DataSignal": false,
                        "DetectedByRequester": false,
                        "DetectedByResponder": false,
                        "ErrorType": {
                            "Description": "Storage error in memory (DRAM).",
                            "Name": "ERR_MEM",
                            "Value": 4
                        },
                        "FirstError": false,
                        "OverflowDroppedLogs": false
                    },
                    "Extended": {
                        "ChipIdentification": 0,
                        "RowBit16": false,
                        "RowBit17": false
                    },
                    "MemoryErrorType": {
                        "Name": "Scrub Uncorrected Error",
                        "Value": 14
                    },
                    "ModuleRank": 0,
```

```
                    "ModuleSmbiosHandle": 0,
                    "Node": 0,
                    "PhysicalAddress": 12884901888,
                    "PhysicalAddressMask": 281474976710655,
                    "RankNumber": 0,
                    "RequestorID": 0,
                    "ResponderID": 0,
                    "Row": 40960,
                    "TargetID": 0,
                    "ValidationBits": {
                        "BankAddressValid": false,
                        "BankGroupValid": true,
                        "BankValid": true,
                        "BitPositionValid": true,
                        "CardHandleValid": false,
                        "CardValid": false,
                        "ChipIdentificationValid": false,
                        "ColumnValid": true,
                        "DeviceValid": false,
                        "ErrorStatusValid": true,
                        "ExtendedRowBitsValid": true,
                        "MemoryErrorTypeValid": true,
                        "MemoryPlatformTargetValid": false,
                        "ModuleHandleValid": false,
                        "ModuleValid": true,
                        "NodeValid": false,
                        "PhysicalAddressMaskValid": true,
                        "PhysicalAddressValid": true,
                        "PlatformRequestorIDValid": false,
                        "PlatformResponderIDValid": false,
                        "RankNumberValid": true,
                        "RowValid": true
                    }
                }
            }
        },
        "SectionType": "a5bc1114-6f64-4ede-b863-3e83ed7c83b1"
    },
    "Created": "2024-11-15T10:40:08+00:00",
    "DiagnosticDataType": "CPERSection",
    "EntryType": "Event",
    "Id": "6",
    "Message": "A platform error occurred.",
    "MessageArgs": [],
    "MessageId": "Platform.1.0.PlatformError",
    "Name": "System Event Log Entry",
    "Resolution": "Check additional diagnostic data if available.",
    "Resolved": false,
    "Severity": "Warning"
}
```


7.6.2.2.2.3 RAS PCIe Error


```
{
    "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/15",
    "@odata.type": "#LogEntry.v1_15_0.LogEntry",
    "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/15/attachment",
    "CPER": {
        "NotificationType": "cf93c01f-1a16-4dfc-b8bc-9c4daf67c104",
        "Oem": {
            "Nvidia": {
                "@odata.type": "#NvidiaCPER.v1_0_0.NvidiaCPER",
                "Pcie": {
                    "AerInfo": {
                        "Capabilites_control": 0,
                        "Capability_header": 0,
                        "Correctable_error_mask": 0,
                        "Correctable_error_status": 0,
                        "Correctable_error_status_hex": "0x00000000",
                        "Data":
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAA",
                        "Tlp_header_0": 0,
                        "Tlp_header_1": 0,
                        "Tlp_header_2": 0,
                        "Tlp_header_3": 0,
                        "Uncorrectable_error_mask": 0,
                        "Uncorrectable_error_severity": 0,
                        "Uncorrectable_error_status": 0,
                        "Uncorrectable_error_status_hex": "0x00000000"
                    },
                    "BridgeControlStatus": {
                        "ControlRegister": 0,
                        "SecondaryStatusRegister": 0
                    },
                    "CapabilityStructure": {
                        "Data": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
                    },
                    "CommandStatus": {
                        "CommandRegister": 0,
                        "StatusRegister": 0
                    },
                    "DeviceID": {
                        "ClassCode": 0,
                        "DeviceID": 5555,
                        "DeviceIDHex": "0x15B3",
                        "DeviceNumber": 0,
                        "FunctionNumber": 0,
                        "PrimaryOrDeviceBusNumber": 0,
```

```
                    "SecondaryBusNumber": 0,
                    "SegmentNumber": 0,
                    "SlotNumber": 0,
                    "VendorID": 41692
                },
                "DeviceSerialNumber": 0,
                "PortType": {
                    "Name": "Unknown",
                    "Value": 16777216
                },
                "ValidationBits": {
                    "AerInfoValid": false,
                    "BridgeControlStatusValid": false,
                    "CapabilityStructureStatusValid": false,
                    "CommandStatusValid": false,
                    "DeviceIDValid": false,
                    "DeviceSerialNumberValid": false,
                    "PortTypeValid": false,
                    "VersionValid": false
                },
                "Version": {
                    "Major": 0,
                    "Minor": 0
                }
            }
        }
    },
    "SectionType": "d995e954-bbc1-430f-ad91-b44dcb3c6f35"
},
"Created": "2025-02-12T12:09:42+00:00",
"DiagnosticDataType": "CPERSection",
"EntryType": "Event",
"Id": "15",
"Message": "A platform error occurred.",
"MessageArgs": [],
"MessageId": "Platform.1.0.PlatformError",
"Name": "System Event Log Entry",
"Resolution": "Check additional diagnostic data if available.",
"Resolved": false,
"Severity": "OK"
}
```


7.6.2.2.2.4 ATX Power Error


```
{
    "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/3",
    "@odata.type": "#LogEntry.v1_15_0.LogEntry",
    "CPER": {
        "NotificationType": "6d5244f2-2712-11ec-bea7-cb3fdb95c786",
        "Oem": {
            "Nvidia": {
                "@odata.type": "#NvidiaCPER.v1_0_0.NvidiaCPER",
                "Nvidia": {
                    "ErrorInstance": 0,
                    "ErrorType": 4,
                    "InstanceBase": 0,
                    "RegisterCount": 1,
                    "Registers": [
                        {
                            "Address": 0,
                            "Value": 1
                        }
                    ],
                    "Severity": {
                        "Code": 1,
                        "Name": "Fatal"
                    },
                    "Signature": "NBU",
                    "Socket": 0
                }
            }
        },
        "SectionType": "6d5244f2-2712-11ec-bea7-cb3fdb95c786"
    },
    "Created": "2025-01-16T08:37:32+00:00",
    "DiagnosticData": "Q1BFUgAD/////
wEAAQAAAAAAAAAD4AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADyRFJtEifsEb6nyz/
blceGIwAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADIAAAAMAAAAAAAADAAAAAAAAAA8kRSbRIn7BG+p8s/
25XHhgAAAAAAAAAAAAAAAAAAAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABOQlUAAAAAAAAAAAAAAAAABAAAAAEAAQAAAAAAAAAAAAAAAAAAAAAAAAAQA
AAAAAAAA",
    "DiagnosticDataType": "CPERSection",
    "EntryType": "Event",
    "Id": "3",
    "Links": {
        "OriginOfCondition": {
            "@odata.id": "/redfish/v1/Systems/Bluefield/Processors/CPU_0"
        }
    },
    "Message": "A platform error occurred.",
    "MessageArgs": [],
    "MessageId": "Platform.1.0.PlatformError",
    "Name": "System Event Log Entry",
    "Resolution": "Check additional diagnostic data if available.",
    "Resolved": false,
    "Severity": "Critical"
}
```

7.6.3 BMC Sensor Data

7.6.3.1 SDR Sensor List

The following is a list of the available sensors maintained by the BMC including their type and name.

Sensor Name	Sensor Type	Source	Description
`p0_link`	Discrete	IPMB	Uplink port 0 link status • 0x100 – connection OK • 0x200 – connection error
`p1_link`	Discrete	IPMB	Uplink port 1 link status • 0x100 – connection OK • 0x200 – connection error
`bluefield_temp`	Temperature	IPMB	NVIDIA® BlueField® temperature
`p0_temp`	Temperature	IPMB	Uplink port 0 SFP temperature
`p1_temp`	Temperature	IPMB	Uplink port 1 SFP temperature
`ddr_temp`	Temperature	IPMB	DDR temperature
`rtc_voltage`	Voltage	IPMB	RTC battery voltage
`power_envelope`	Power	IPMB	• This sensor indicates the maximum power consumption allowed for BlueField-3 DPU • This sensor is incompatible with secured Linux
`soc_power`	Power	IPMB	• This sensor indicates the current power consumption of the BlueField-3 DPU • This sensor is incompatible with secured Linux
`power_envelope_deviation`	Power	Synthesized Sensor	Measures the deviation of `soc_power` sensor value from `power_envelope` sensor value. • `power_envelope_deviation = soc_power - power_envelope` • The sensor value should be negative for normal conditions. If the sensor value is positive, then SoC power has exceeded the allowed power envelope. • If the value of `soc_power` or `power_envelope` sensors is NaN, then the `power_envelope_deviation` sensor value will also be NaN.
`1V_BMC`	Voltage	BMC ADC	
`1_2V_BMC`	Voltage	BMC ADC	
`1_8V`	Voltage	BMC ADC	
`1_8V_BMC`	Voltage	BMC ADC	
`2_5V`	Voltage	BMC ADC	
`3_3V`	Voltage	BMC ADC	

Sensor Name	Sensor Type	Source	Description
3_3V_RGM	Voltage	BMC ADC	
5V	Voltage	BMC ADC	
12V_ATX	Voltage	BMC ADC	Input power rail from ATX (power from gold fingers in case of Sub75 when ATX power is off)
12V_PCIe	Voltage	BMC ADC	Input power rail from gold fingers
DVDD	Voltage	BMC ADC	
HVDD	Voltage	BMC ADC	
VDD	Voltage	BMC ADC	
VDDQ	Voltage	BMC ADC	
VDD_CPU_L	Voltage	BMC ADC	
VDD_CPU_R	Voltage	BMC ADC	

IPMB sourced sensors are supported when operating in DPU mode only.

7.6.3.2 Sensor Redfish Commands

7.6.3.2.1 Getting List of Support Sensors

BlueField sensors are stored within the Sensors schema under the Chassis schema. To retrieve the list of supported sensors, execute the following command:

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

The following is an example of the anticipated output:

```
{
  "@odata.id": "/redfish/v1/Chassis/Card1/Sensors",
  "@odata.type": "#SensorCollection.SensorCollection",
  "Description": "Collection of Sensors for this Chassis",
  "Members": [
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/power_envelope"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/power_envelope_deviation"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/soc_power"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/bluefield_temp"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/ddr_temp"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/p0_temp"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/p1_temp"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/12V_ATX"
    },
```

```
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/12V_PCIe"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/1V_BMC"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/1_2V_BMC"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/1_8V"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/1_8V_BMC"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/2_5V"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/3_3V"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/3_3V_RGM"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/5V"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/DVDD"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/HVDD"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/VDD"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/VDDQ"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/VDD_CPU_L"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/VDD_CPU_R"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/rtc_voltage"
    }
  ],
  "Members@odata.count": 24,
  "Name": "Sensors"
}
```

7.6.3.2.2 Getting Data for Specific Sensor

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

The following is an example of a temperature sensor BlueField reading:

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET https://<bmc_ip>/redfish/v1/Chassis/Card1/
Sensors/bluefield_temp
{
  "@odata.id": "/redfish/v1/Chassis/Card1/Sensors/bluefield_temp",
  "@odata.type": "#Sensor.v1_2_0.Sensor",
  "Id": "bluefield_temp",
  "Name": "bluefield temp",
  "Reading": 43.0,
  "ReadingRangeMax": 255.0,
  "ReadingRangeMin": 0.0,
  "ReadingType": "Temperature",
  "ReadingUnits": "Cel",
  "RelatedItem": [
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield"
    }
  ],
  "Status": {
    "Conditions": [],
    "Health": "OK",
    "HealthRollup": "OK",
    "State": "Enabled"
  },
  "Thresholds": {
    "LowerCaution": {
      "Reading": 5.0
    },
    "LowerCritical": {
      "Reading": 0.0
    },
    "UpperCaution": {
      "Reading": 95.0
    },
    "UpperCritical": {
```

```
      "Reading": 105.0
    }
  }
}
```

7.6.3.2.3 Configuring Sensor Thresholds

The following commands set the thresholds for sensors that support setting a threshold:

```
curl -k -u root:'<password>' -X PATCH https://<bmc_ip>/redfish/v1/Chassis/Card1/Sensors/<sensor name>/ -d
'{"Thresholds":{"<Threshold name>": {"Reading":<value>}}}'
```

The following is an example of how to set the upper critical threshold for the BlueField temperature sensor:

```
curl -k -u root:'<password>' -X PATCH https://<bmc_ip>/redfish/v1/Chassis/Card1/Sensors/bluefield_temp -d
'{"Thresholds":{"UpperCritical": {"Reading":100}}}'
{
  "@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"
    }
  ]
}
```

7.6.3.3 Sensor IPMI Commands

BMC software supports reading chassis sensor information using the IPMItool.

The following subsections list commands which allow reading SDR data.

7.6.3.3.1 Displaying Sensor Data

Displays sensor data repository entry readings and their status.

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

7.6.3.3.2 Displaying Extended Sensor Data

Displays extended sensor information.

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

7.6.3.3.3 Displaying Sensors and Thresholds

Displays sensors and thresholds in a wide table format.

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

7.6.3.3.4 Displaying Sensor Data Records Specified by Sensor ID

Displays sensor data records specified by sensor ID.

```
ipmitool -C 17 -I lanplus -H <bmc_ip> -U ADMIN -P ADMIN sdr get <name>
```

7.6.3.3.5 Displaying All Records from SDR Repository of Specific Type

Displays all records from the SDR repository of a specific type.

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

7.6.3.3.6 Displaying Data for Sensors Specified by Name

Displays information for sensors specified by name.

```
ipmitool -C 17 -I lanplus -H <bmc_ip> -U ADMIN -P ADMIN sensor get <sensor_name>
```

7.6.3.3.7 Displaying Readings for Sensors Specified by Name (Only for Numeric Sensors)

```
ipmitool -C 17 -I lanplus -H <bmc_ip> -U ADMIN -P ADMIN sensor reading <name>…<name>
```

7.6.4 BlueField Arm State

This section outlines methods for monitoring the state of NVIDIA® BlueField® Arm using either Redfish or IPMI.

> The BMC polls the host status from the NIC subsystem on BlueField using the NC-SI interface approximately every 30 seconds. Due to this implementation, some stages of the Arm boot process may not be captured. The expected `OemLastState` values to indicate boot completion for the different modes are as follows:
>
> - `OsIsRunning` - for DPU mode
> - `UEFI` - for NIC mode

7.6.4.1 Monitoring BlueField Arm State Using Redfish

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

The BlueField Arm state is represented by the `OemLastState` field under `BootProgress`.

Example output:

```
...
"BootProgress": {
    ...
    "OemLastState": "OsIsRunning"
  }
...
```

The possible values for `OemLastState` are:

- `BootRom`
- `BL2`
- `BL31`
- `UEFI`
- `OsStarting`
- `OsIsRunning`
- `LowPowerStandby`
- `FirmwareUpdateInProgress`
- `OsCrashDumpInProgress`
- `OsCrashDumpIsComplete`
- `FWFaultCrashDumpInProgress`
- `FWFaultCrashDumpIsComplete`
- `Invalid`

7.6.4.2 Monitoring BlueField Arm State Using IPMI

To get the BlueField Arm state with IPMI, refer to the `0xA3` command under "IPMItool NIC Subsystem Management".

7.6.5 Rsyslog

It is possible to dynamically configure rsyslog servers to receive system event log (SEL) messages and/or the BlueField SoC UART console printout (SOL) messages.

7.6.5.1 SEL and SOL Message Reception Format

SEL messages are received on the rsyslog server in the following format:

```
<Timestamp> <host> <EntryID-hex> | <Date> | <Time> | <Sensor-Type> | <Event-Type> | <Event-Direction> |
<Description>
```

For example:

```
"2024-06-18T11:05:45.926095+03:00 ldev-platform-12-244.exam 75 | 06/18/24 | 08:05:45 UTC | Voltage #0x08 | Lower
Non-critical going low | Asserted"
```

SOL messages are received on the rsyslog server exactly as they appear in the BlueField console, including a timestamp and the hostname:

```
<Timestamp> <host> <message>
```

For example:

```
"2024-06-18T15:16:28.240538+03:00 ldev-platform-12-244 systemd[1]: Starting RDMA Node Description Daemon"
```

⚠ `$EscapeControlCharactersOnReceive` and `$Escape8BitCharactersOnReceive` should be turned off on the rsyslog server side.

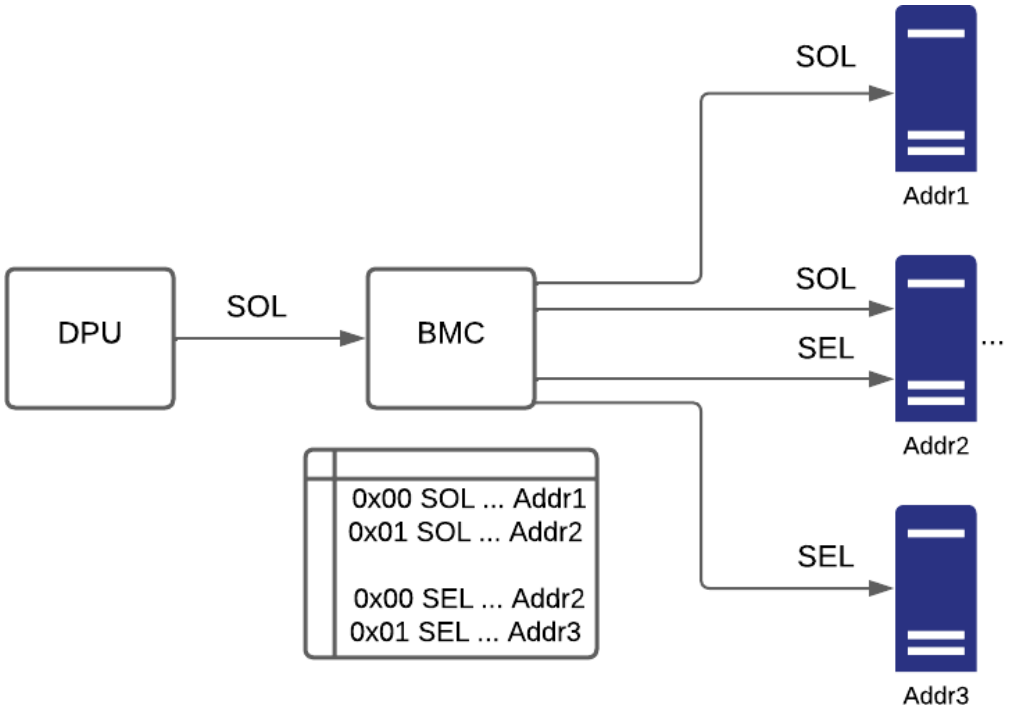
7.6.5.2 Rsyslog Servers Configurations

The rsyslog configurations define data streams. Each of them includes:

- Configuration identifier – An index (ranging from 0x00 to 0x09) AND a log type (SEL 0x01 or SOL 0x03)
- Status – Enable/disable
- Transport protocol – TCP/UDP
- Network protocol – IPv4/IPv6
- Server address
- Port

Note that configurations with the same index but different log types are considered to be different configurations. For example, 0x01-SOL and 0x01-SEL are distinct configurations.

The following diagram illustrates an example of three rsyslog servers receiving four data streams:



This setup requires four configurations:

- Configuration 0x00-SOL – Server1 receives SOL
- Configuration 0x01-SOL – Server2 receives SOL
- Configuration 0x00-SEL – Server2 receives SEL

- Configuration 0x01-SEL – Server3 receives SEL

> ⚠ The BMC rsyslog configuration files located under `/etc/rsyslog.d` are automatically generated and are read-only. These files can only be modified using the IPMI commands listed later on this page.

7.6.5.2.1 IPMI Commands

The following table lists the IPMI commands for setting and getting rsyslog servers configurations:

netfunc	cmd	data	Description
`0x32`	`0xD3`	`<Index> <LogType>`	Get rsyslog status – Displays information of the configured rsyslog server The request contains the index and the log type of the rsyslog server configuration, and it is 2 bytes long. The response contains the following information: `<Index> <LogType> <Status> <TransportProtocol> <NetworkProtocol> <ServerAddress> <Port>` • Byte 1 – Completion code: • 0x00 – Success (does not appear in IPMI textual response) • 0x01 – Failure (the rest does not appear in IPMI response) • Byte 2 – Index • Index of server (0x00-0x09) • Byte 3 – LogType • 0x01 – SEL • 0x03 – SOL • Byte 4 – Status • 0x00 – Disabled • 0x01 – Enabled • Byte 5 – Transport protocol • 0x00 – UDP • 0x01 – TCP • Byte 6 – Network protocol • 0x00 – IPv4 • 0x01 – IPv6 • Byte 7-n – Rsyslog server address • Rsyslog addr (4/16 Bytes) • Byte n+1-n+2 – Port • Rsyslog port. LSB first. The response is 12 bytes long for IPv4 and 24 bytes long for IPv6.

netfunc	cmd	data	Description
`0x32`	`0xD4`	`<Index> <LogType> <Status> <TransportProtocol> <NetworkProtocol> <ServerAddress> <Port>`	Set rsyslog status - • Configures a new rsyslog server configuration if the configuration `<Index> <LogType>` does not exist. • Modifies an existing rsyslog server configuration if the configuration `<Index> <LogType>` does exist. The command contains the following information: • Byte 1 - Index • Index of server (0x00-0x09) • Byte 2 - LogType • 0x01 - SEL • 0x03 - SOL • Byte 3 - Status • 0x00 - Disabled • 0x01 - Enabled • Byte 4 - Transport protocol • 0x00 - UDP • 0x01 - TCP • Byte 5 - Network protocol • 0x00 - IPv4 • 0x01 - IPv6 • Byte 6-n - Rsyslog server address • Rsyslog addr (4/16 Bytes) • Byte n+1-n+2 - Port • Rsyslog port. LSB first. The command data is 11 bytes long for of IPv4 and 23 bytes log for IPv6. The response contains the completion code and is 1 byte long. The success completion code does not appear in IPMI textual response.

7.6.5.2.2 Usage Examples

7.6.5.2.2.1 Setting Rsyslog Status of Two Configurations

The following commands create or modify two different rsyslog configurations with Index 0x00 and LogTypes SEL/SOL :
netfunc: 0x32, cmd: 0xD4, Indx: 0x00, LogType: 0x01(SEL) / 0x03(SOL), status: 0x01 (Enabled), TP: 0x01 (TCP), NP: 0x00 (IPv4) Address: 0x0A 0xED 0x33 0xF4 (10.237.51.244) Port: 0xFA 0x13 (5114)

```
root@dpu-bmc:~# ipmitool raw 0x32 0xD4 0x00 0x01 0x01 0x01 0x00 0x0A 0xED 0x33 0xF4 0xFA 0x13
root@dpu-bmc:~# ipmitool raw 0x32 0xD4 0x00 0x03 0x01 0x01 0x00 0x0a 0xed 0x33 0xf4 0xfa 0x13
```

Now the same rsyslog server receives both SEL and SOL messages.

The following command disables the rsyslog configurations with Index 0x00 and LogTypes SOL:
netfunc: 0x32, cmd: 0xD4, Indx: 0x00, LogType: 0x03 (SOL), status: 0x00 (Disabled), TP: 0x01 (TCP), NP: 0x00 (IPv4) Address: 0x0A 0xED 0x33 0xF4 (10.237.51.244) Port: 0xFA 0x13 (5114)

```
root@dpu-bmc:~# ipmitool raw 0x32 0xD4 0x00 0x03 0x00 0x01 0x00 0x0A 0xED 0x33 0xF4 0xFA 0x13
```

Now the rsyslog server receives only SEL messages as the SOL configuration is disabled:
netfunc: 0x32, cmd: 0xD3, Indx: 0x00, LogType: 0x01(SEL) / 0x03(SOL)

```
root@dpu-bmc:~# ipmitool raw 0x32 0xD3 0x00 0x01
 00 01 01 01 00 0a ed 33 f4 fa 13
root@dpu-bmc:~# ipmitool raw 0x32 0xD3 0x00 0x03
 00 03 00 01 00 0a ed 33 f4 fa 13
```

7.6.5.2.2.2 Setting Rsyslog Status with IPv6 Address

The following command creates or modified an rsyslog configuration with an IPv6 address:
netfunc: 0x32, cmd: 0xD4, Indx: 0x07, LogType: 0x01 (SEL), status: 0x01 (Enabled), TP: 0x01 (TCP), NP: 0x01 (IPv6) Address: 0xFD 0xFD 0xFD 0xFD 0x00 0x10 0x02 0x37 0x02 0x50 0x56 0xFF 0xFE 0x30 0x33 0xF4 (FDFD:FDFD:10:237:250:56FF:FE30:33F4) Port: 0xFA 0x13 (5114)

```
root@dpu-bmc:~# ipmitool raw 0x32 0xD4 0x07 0x01 0x01 0x01 0x01 0xfd 0xfd 0xfd 0xfd 0x00 0x10 0x02 0x37 0x02 0x50
0x56 0xff 0xfe 0x30 0x33 0xf4 0xfa 0x13
```

7.6.5.2.2.3 Setting Rsyslog Status with Invalid Argument

The following command attempts to create an rsyslog server configuration with an invalid index 0x0A (Valid indexes are 0x00-0x09):
netfunc: 0x32, cmd: 0xD4, Indx: 0x0A, LogType: 0x01 (SEL), status: 0x01 (Enabled), TP: 0x01 (TCP), NP: 0x00 (IPv4) Address: 0x0A 0xED 0x33 0xF4 (10.237.51.244) Port: 0xFA 0x13 (5114)

```
root@dpu-bmc:~# ipmitool raw 0x32 0xD4 0x0A 0x01 0x01 0x01 0x00 0x0A 0xED 0x33 0xF4 0xFA 0x13
Unable to send RAW command (channel=0x0 netfn=0x32 lun=0x0 cmd=0xd4 rsp=0xcc): Invalid data field in request
```

7.6.5.2.2.4 Getting Rsyslog Status Information

The following command displays the information of the rsyslog configuration with index 0 and LogType SEL :
netfunc: 0x32, cmd: 0xD3, Indx: 0x00, LogType: 0x01(SEL)

```
root@dpu-bmc:~# ipmitool raw 0x32 0xD3 0x00 0x01
 00 01 01 01 00 0a ed 33 f4 fa 13
```

7.6.5.2.2.5 Getting Non-existing Rsyslog Server Information

The following command attempts to receive an information of a non-existing rsyslog configuration with index 0x06 and LogType SEL :
netfunc: 0x32, cmd: 0xD3, Indx: 0x06, LogType: 0x01(SEL)

```
root@dpu-bmc:~# ipmitool raw 0x32 0xD3 0x06 0x01
Unable to send RAW command (channel=0x0 netfn=0x32 lun=0x0 cmd=0xd3 rsp=0xcc): Invalid data field in request
```

7.6.6 DPU Chassis

The Redfish `Chassis` schema provides a structured and standardized way to represent essential information about the physical infrastructure of computing systems (i.e., the NVIDIA® BlueField®),

offering valuable insights for system administrators, data center operators, and management software developers.

The BlueField 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"
}
```

7.6.6.1 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": ""
}
```


7.6.6.2 Chassis Card1 NetworkAdapters

> ⚠ Retrieving these values is supported when operating in DPU mode only.

The `NetworkAdapters` schema specifically aims to standardize NIC management and representation. This schema includes a collection under `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/Ports/eth0
  ```

  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/NetworkDeviceFunctions/eth0f0
  ```

  Example output:


  ```
  {
    "@odata.id": "/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/NetworkDeviceFunctions/
  eth0f0",
    "@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"
}
```


⚠ Removing or adding new ports requires a BMC reboot.

7.6.7 DPU Information

⚠ The following actions are supported when operating in DPU mode only.

7.6.7.1 Getting Base GUID

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

7.6.7.2 Getting Base MAC

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

7.6.7.3 Getting Description

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

7.6.8 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.

7.6.8.1 BMC Dump Operations

The following subsections list BMC dump operations.

7.6.8.1.1 Creating BMC Dump Task

Create a BMC dump task and gets the task ID.

> 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/LogService.CollectDiagnosticData
```

Where:

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

> This command triggers an attempt to enable the RShim on the BMC.

> Notes about the size of a single BMC dump and the BMC dumps container:
>
> - The total size of all BMC dumps cannot exceed 8MB
> - A single BMC dump cannot take up more than 4MB. If it is larger, it is truncated to 4MB.
> - For the proper creation of a BMC dump, 4MB of free memory are required regardless of its actual size (can be smaller than 4MB). This memory is ensured by deleting existing BMC dumps, from oldest to newest, until 4MB are free.

7.6.8.1.2 Getting 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

7.6.8.1.3 Downloading 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/<entry_id>/attachment --output </path/to/tar/log_dump.zstd>
```

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.zstd>` - path to download the log dump `log_dump.zstd`

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

Log list:

- `journal-pretty.log`
- `sensor-readings.log`
- `host-state.log`
- `hostnamectl.log`
- `fw-version.log`
- `fru-info.log`
- `nicDeviceDebugInfo.log (mstdump)`
  - CRspace and Scartchpad address spaces
  - Dumps are lists of 32-bit addresses and the 32-bit values stored at these addresses
  - Only works if NIC is working
  - Only on BlueField-3
- `chassis-state.log`
- `bmc-state.log`
- `rshim.log`
- `uptime.log`
- `cpuinfo`
- `fw-printenv.log`
- `varfilelist.log`
- `tmpfilelist.log`
- `softIRQs.log`
- `sensorinfo.log`
- `selinfo.log`
- `pslist.log`
- `routeinfo.log`
- `network.log`
- `network`
  - `00-bmc-eth0.network`
  - `00-tmfifo_net0.network`
  - `00-bmc-vlan4040.network`
  - `vlan4040.netdev`
- `netstat.log`
- `mntinfo.log`
- `kernalcmdline.log`
- `kernalRingBuff.log`

- `iproute.log`
- `dpulogs`
    - `dpu_console`
    - `obmc-console.log`
- `iplink.log`
- `ipaddr.log`
- `interrupts.log`
- `freemem.log`
- `channelconfig.log`
- `channelaccess.log`
- `arptable.log`
- `inventory.log`
- `elogall.log`
- `os-release`
- `top.log`
- `meminfo`
- `failed-services.log`
- `hwmon.log`
- `tmpfilelist.log`
- `slabinfo.log`
- `settings.log`
- `dmesg.log`
- `em-system.json`
- `dmesg.log`
- `bios.log`
- `timedate.log`
- `lsof.log`
- `dreport.log`
- `disk-usage.log`
- `summary.log`

To access logs under `bflogs`, BlueField would have to operate in DPU mode.

7.6.8.1.4 Deleting All Dump Entries

Clear all log dump entries.

```
sudo curl -k -u root:'<password>' -H 'Content-Type: application/json' -X POST 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.

7.6.8.2 System Dump Operations

The following subsections list system dump operations.

7.6.8.2.1 Creating 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/LogService.CollectDiagnosticData
```

Where:

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

Note, this command triggers an attempt to enable the RShim on the BMC.

7.6.8.2.2 Getting 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

7.6.8.2.3 Downloading 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_id>/attachment --output </path/to/tar/system_dump.zstd>
```

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.zstd>` - path to download the log dump `system_dump.zstd`

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

Dump list:

- `bflogs/dmesg`
- `bflogs/lastlog`
- `bflogs/wtmp`

  > To access logs under `bflogs`, BlueField would have to operate in DPU mode.

- `rshim.log`
- `dreport.log`
- `summary.log`

7.6.8.2.4 Deleting All Dump Entries

Clear all log dump entries.

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

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.

7.6.8.3 BlueField Console Log

BMC captures the BlueField console output and stores it in the System dump. Refer to section "System 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.

7.6.9 System Processor

> ⚠ System processor information is sourced from the UEFI and stored in the BMC's persistent memory. This data may be lost after a BMC factory reset and will be restored upon the next UEFI reboot.

7.6.9.1 Processor Summary

A high-level summary of the BlueField processors is available in the `ProcessorSummary` object within the Redfish `ComputerSystem` schema.

To retrieve this information, run the following command:

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

Example output:

```
{
    ...,
    "@odata.id": "/redfish/v1/Systems/Bluefield",
    "@odata.type": "#ComputerSystem.v1_22_0.ComputerSystem",
    ...,
    "ProcessorSummary": {
        "CoreCount": 16,
        "Count": 1,
        "Model": "ARMv8"
    },
    "Processors": {
        "@odata.id": "/redfish/v1/Systems/Bluefield/Processors"
    },
    ...
}
```

The `ProcessorSummary` provides:

- `CoreCount` - Total number of cores across all processors.
- `Count` - Total number of processors.
- `Model` - Processor model of the primary processor.

7.6.9.2 Processor Collection

A detailed list of system processors is available in the `ProcessorCollection` object.

To retrieve the processor collection, run:

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

Example output:

```
{
    "@odata.id": "/redfish/v1/Systems/Bluefield/Processors",
    "@odata.type": "#ProcessorCollection.ProcessorCollection",
    "Members": [
        {
            "@odata.id": "/redfish/v1/Systems/Bluefield/Processors/CPU_0"
        }
    ],
    "Members@odata.count": 1,
```

```
    "Name": "Processor Collection"
}
```

Each entry in the `Members` array represents a processor in the system. In this example, the collection contains a single processor: `CPU_0`.

7.6.9.3 Individual Processor Information

Detailed information about an individual processor is provided in the Redfish `Processor` schema.

To retrieve information for `CPU_0`, run the following command:

```
curl -k -u <bmc_username>:<bmc_password> -H 'Content-Type: application/json' -X GET https://<bmc_ip>/redfish/v1/
Systems/Bluefield/Processors/CPU_0
```

Example output:

```
{
    "@Redfish.Settings": {
        "@odata.type": "#Settings.v1_3_3.Settings",
        "SettingsObject": {
            "@odata.id": "/redfish/v1/Systems/Bluefield/Processors/CPU_0/Settings"
        }
    },
    "@odata.id": "/redfish/v1/Systems/Bluefield/Processors/CPU_0",
    "@odata.type": "#Processor.v1_20_0.Processor",
    "EnvironmentMetrics": {
        "@odata.id": "/redfish/v1/Systems/Bluefield/Processors/CPU_0/EnvironmentMetrics"
    },
    "Id": "CPU_0",
    "Links": {
        "Chassis": {
            "@odata.id": "/redfish/v1/Chassis/Card1"
        }
    },
    "Location": {
        "PartLocation": {
            "ServiceLabel": "Socket 0"
        }
    },
    "Manufacturer": "https://www.mellanox.com",
    "MaxSpeedMHz": 2135,
    "Metrics": {
        "@odata.id": "/redfish/v1/Systems/Bluefield/Processors/CPU_0/ProcessorMetrics"
    },
    "Model": "Mellanox BlueField-3 [A1] A78(D42) 16 Cores r0p1",
    "Name": "Processor",
    "PartNumber": "OPN: 9009D3B600CVAA",
    "Ports": {
        "@odata.id": "/redfish/v1/Systems/Bluefield/Processors/CPU_0/Ports"
    },
    "ProcessorId": {
        "EffectiveFamily": "0x0101",
        "IdentificationRegisters": "0x00000000410FD421"
    },
    "ProcessorType": "CPU",
    "SerialNumber": "Unspecified Serial Number",
    "Socket": "Socket 0",
    "Status": {
        "Conditions": [],
        "Health": "OK",
        "State": "Enabled"
    },
    "TotalCores": 16,
    "TotalThreads": 16,
    "Version": "Mellanox BlueField-3 [A1] A78(D42) 16 Cores r0p1"
}
```

7.6.9.3.1 Supported Properties

Property	Type	Description
`Id`	string	Unique identifier of the processor.
`MaxSpeedMHz`	integer (MHz)	Maximum clock speed of the processor.
`Model`	string	Processor model number (matches the `Version` property).

Property	Type	Description
`Name`	string	Name of the processor.
`PartNumber`	string	Part number assigned to the processor.
`ProcessorType`	string	Type of processor (e.g., CPU, GPU).
`Status`	object	Status object reporting health and operational state.
`TotalCores`	integer	Total number of cores in the processor.
`TotalThreads`	integer	Total number of execution threads supported by the processor.
`Version`	string	Hardware version of the processor.

7.7 Network

This section contains the following pages:

7.7.1 BlueField Host Network Interface

Under URI `redfish/v1/Systems/Bluefield`, there is a collection called `EthernetInterfaces` representing the data ports and the OOB port of the BlueField. It is read-only and contains network information (e.g., IP addresses, MAC addresses).

These abilities are supported when operating in DPU mode only.

7.7.1.1 Displaying Network Interfaces Collection

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET https://<bmc_ip>/redfish/v1/Systems/
Bluefield/EthernetInterfaces
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/EthernetInterfaces",
  "@odata.type": "#EthernetInterfaceCollection.EthernetInterfaceCollection",
  "Description": "Collection of EthernetInterfaces of the host",
  "Members": [
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/EthernetInterfaces/eth0"
    },
    {
      "@odata.id": "/redfish/v1/Systems/Bluefield/EthernetInterfaces/oob0"
    }
  ],
  "Members@odata.count": 2,
  "Name": "Ethernet Network Interface Collection"
}
```

7.7.1.2 Displaying Network Interface Object

The interface object has a field called `LinkStatus` which is determined by the following rules:

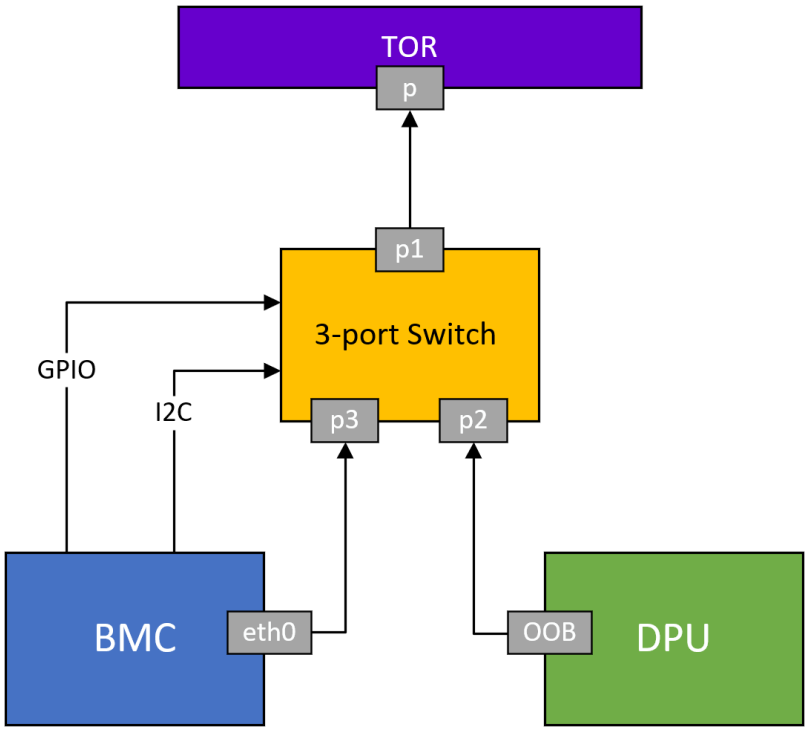
- If the interface is the OOB port (i.e., `oob_net0`), `LinkStatus` would display `LinkUp` if the port is configured up using ifconfig/ip command.
- If the interface is a data port (i.e., `eth0` / `eth1` or `ib0` / `ib1`), `LinkStatus` would display `NoLink` if no QSFP cable is connected. If a QSFP transceiver is connected, the link would appear as `LinkUp` if the port is configured as up using the `ifconfig` / `ip` commands. If not, it displays `LinkDown`.

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET https://<bmc_ip>/redfish/v1/Systems/
Bluefield/EthernetInterfaces/oob0
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/EthernetInterfaces/oob0",
  "@odata.type": "#EthernetInterface.v1_6_0.EthernetInterface",
  "DHCPv4": {
    "DHCPEnabled": true,
    "UseDNSServers": false,
    "UseDomainName": false,
    "UseNTPServers": false
  },
  "DHCPv6": {
    "OperatingMode": "Stateful",
    "UseDNSServers": false,
    "UseDomainName": false,
    "UseNTPServers": false
  },
  "Description": "Host Network Interface for port oob0",
  "IPv4Addresses": [
    {
      "Address": "10.345.41.97",
      "AddressOrigin": "Static",
      "Gateway": "0.0.0.0",
      "SubnetMask": "255.255.240.0"
    }
  ],
  "IPv4StaticAddresses": [
    {
      "Address": "10.345.41.97",
      "AddressOrigin": "Static",
      "Gateway": "0.0.0.0",
      "SubnetMask": "255.255.240.0"
    }
  ],
  "IPv6AddressPolicyTable": [],
  "IPv6Addresses": [
    {
      "Address": "fe80::a278:c2ff:fe0e:87a4",
      "AddressOrigin": "Static",
      "AddressState": null,
      "PrefixLength": 64
    }
  ],
  "IPv6DefaultGateway": "0:0:0:0:0:0:0:0",
  "IPv6StaticAddresses": [
    {
      "Address": "fe80::a278:c2ff:fe0e:87a4",
      "PrefixLength": 64
    }
  ],
  "Id": "oob0",
  "InterfaceEnabled": true,
  "LinkStatus": "LinkUp",
  "MACAddress": "a0:88:a2:0e:87:a4",
  "MTUSize": 1500,
  "Name": "Host Ethernet Interface",
  "NameServers": [],
  "SpeedMbps": 1000,
  "StaticNameServers": [],
  "Status": {
    "State": "Disabled"
  }
}
```

If the user changed the BlueField's IP information dynamically, rebooting the BMC should show the updated IP info.

7.7.2 OOB Network 3-Port Switch Control

To enable both the BMC and the Arm on NVIDIA® BlueField® 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 RJ45 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.



7.7.2.1 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. • 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
`0x32`	`0xA1`	`0x3`	Reset on-board 3-port switch

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
```

7.7.2.2 3-Port Switch Redfish Commands

7.7.2.2.1 Getting 3-port Switch Ports Mode

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

Example output:

```
{
  "LinkStatus": {
    "BMC": "LinkUp",
    "DPU": "LinkUp",
    "RJ45": "LinkUp"
  },
   "TorSwitchMode": {
    "BmcOobEnabled": true,
    "DpuOobEnabled": true
  }
}
```

Where:

- `LinkStatus` - displays the link status of each port on the 3-port switch
  - For the `RJ45` and `DPU` ports, the link status is taken from their `PHY Basic Status Register` and from `PHY Basic Control Register` for the port's power down status on the 3-port switch
  - The BMC link is considered always `LinkUp`
- `TorSwitchMode`:
  - `BmcOobEnabled` - if `true`; enables the BMC to access the out-of-band network
  - `DpuOobEnabled` - if `true`; enables the BlueField to access the out-of-band network

7.7.2.2.2 Setting 3-port Switch Port Mode

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X PATCH -d '{"TorSwitchMode": {"BmcOobEnabled":
<Port State>, "DpuOobEnabled": <Port State>}}' https://<bmc_ip>/redfish/v1/Systems/Bluefield/Oem/Nvidia/Switch
```

Where `Port State`:

- True - Enable the port to access the out-of-band network
- False - Disable the port to access the out-of-band network

⚠ The internal pathway connecting the BMC and RJ45 is not allowed to be disabled using a Redfish command. Therefore, the parameter `BmcOobEnabled` should be set as `true` when setting 3-port switch ports mode, otherwise the Redfish command would return an error.

⚠ For the patch command request, both `BmcOobEnabled` and `DpuOobEnabled` must be set.

The following is an example of how to set only BMC is allowed access to RJ45:

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X PATCH -d '{"TorSwitchMode": {"BmcOobEnabled":
true, "DpuOobEnabled": false}}' https://<bmc_ip>/redfish/v1/Systems/Bluefield/Oem/Nvidia/Switch
```

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"
    }
  ]
}
```

7.7.2.2.3 Resetting On-board 3-port Switch

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X POST https://<bmc_ip>/redfish/v1/Systems/
Bluefield/Oem/Nvidia/Switch.Reset
```

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"
    }
  ]
}
```

7.8 Miscellaneous

This section contains the following topics:

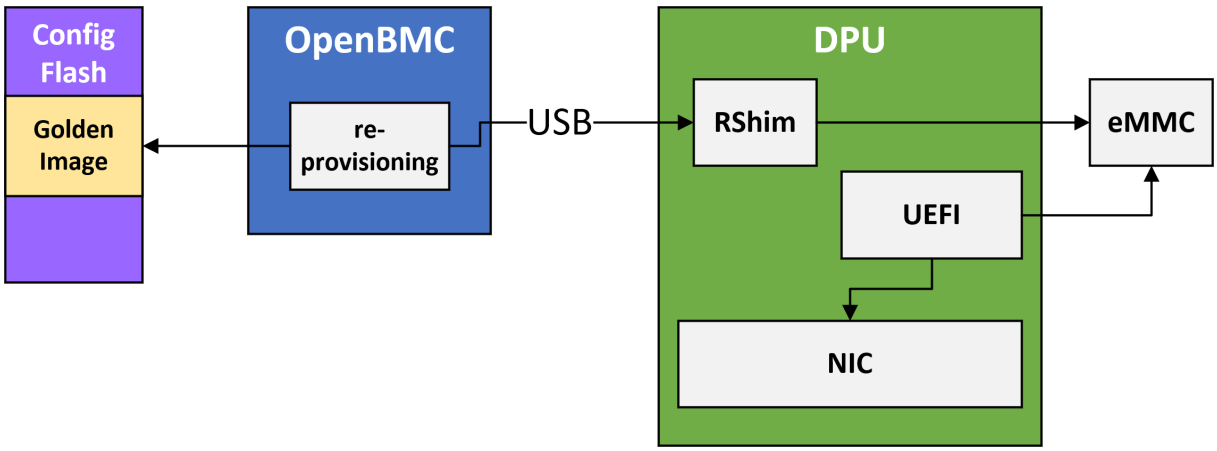
7.8.1 Bare-metal Reprovisioning

> ⚠ Relevant for NVIDIA® BlueField®-3 and later in DPU mode only (not supported in NIC mode).

The re-provisioning flow of the BlueField-3 bare metal system offers a solution for restoring the system to its initial state using built-in resources, eliminating the need for external measures. This approach enables the seamless reloading of the operational image.

To support this functionality, the BMC maintains and manages a golden image for the UEFI and the NIC. This ensures that the UEFI can retrieve the operational image via network protocols such as HTTP or PXE.

The following block diagram provides a high-level overview of the system components and data flow:



The complete flow of network re-provisioning includes the following primary stages:

1. Initial provisioning - Provisioning the golden images to the BMC, typically performed during system manufacturing.
2. In-field updates - Updating the golden images using the in-field update process.
3. OOB network configuration - Configuring network settings through out-of-band (OOB) management.
4. System recovery - Restoring the system by reinstalling the golden images.

7.8.1.1 Initial Provisioning of Golden Image to BMC

1. Ensure the BMC is connected to the out-of-band (OOB) network.
2. Use the `scp` command to transfer the golden images from your local storage to the BMC's temporary storage directories (`/tmp/golden-image-nic/` or `/tmp/golden-image-arm/`).
    - For `golden_image_nic` :

```
#host> scp <nic-golden-image-directory>/<nic-golden-image-filename> root@<bmc-ip>:/tmp/golden-image-nic/
```

- For `golden_image_arm`:

```
#host> scp <arm-golden-image-directory>/<arm-golden-image-filename> root@<bmc-ip>:/tmp/golden-image-arm/
```

> After the image is copied to the BMC's volatile memory, the version is extracted and stored to enable specific features.

> The NIC firmware version is extracted from the image filename. Ensure the filename follows the standard format of official releases.

3. Log into the BMC and use the `dpu_golden_image` utility to transfer the golden images from temporary storage to the BMC's non-volatile storage.
   - For `golden_image_nic`:

```
#bmc> dpu_golden_image golden_image_nic -w /tmp/golden-image-nic/<nic-golden-image-filename>
```

   - For `golden_image_arm`:

```
#bmc> dpu_golden_image golden_image_arm -w /tmp/golden-image-arm/<arm-golden-image-filename>
```

> If the version of the candidate image matches the one already in non-volatile storage, the update is skipped, and the following message is displayed:
>
> ```
> Updating image in memory is cancelled as version will remain unchanged. Force update by adding -f|--force to the command line.
> ```
>
> To force the update, use the `--force` flag:
>
> - For `golden_image_nic`:
>
> ```
> #bmc> dpu_golden_image golden_image_nic -w /tmp/golden-image-nic/<nic-golden-image-filename> --force
> ```
>
> - For `golden_image_arm`:
>
> ```
> #bmc> dpu_golden_image golden_image_arm -w /tmp/golden-image-arm/<arm-golden-image-filename> --force
> ```

4. After provisioning, verify the correctness of the golden images using the following commands:
   - For `golden_image_nic`:

```
#bmc> dpu_golden_image -v golden_image_nic
bmc> echo $?
```

   Expected output: `0`.
   - For `golden_image_arm`:

```
bmc> dpu_golden_image -v golden_image_arm
bmc> echo $?
```

Expected output: `0`.

7.8.1.2 Retrieving Golden Image Version Information

> ⚠ This feature is available only for golden images installed following the upgrade of the BMC firmware to version 24.07-14 or later.

To get the human-readable version (`MAJOR.MINOR.PATCH.BUILD` versioning scheme) of the golden images:

- For `golden_image_nic`:

```
bmc> dpu_golden_image golden_image_nic -V -H
```

- For `golden_image_arm`:

```
bmc> dpu_golden_image golden_image_arm -V -H
```

To get the `sha256sum` value:

- For `golden_image_nic`:

```
bmc> dpu_golden_image golden_image_nic -V
```

- For `golden_image_arm`:

```
bmc> dpu_golden_image golden_image_arm -V
```

7.8.1.3 Retrieving Golden Image Version Information Using Redfish

> ⚠ This feature is available only for Golden Images installed following the upgrade of the BMC firmware to version 24.07-14 or later.

To get the human-readable version (`MAJOR.MINOR.PATCH.BUILD` versioning scheme) of the golden images over the Redfish interface, run:

- For Arm golden image:

```
curl -k -u '<username>':'<password>' -H 'Content-type: application/json' -X GET 'https://<bmc_ip>/redfish/
v1/UpdateService/FirmwareInventory/golden_image_arm'
```

- For NIC golden image:

```
curl -k -u '<username>':'<password>' -H 'Content-type: application/json' -X GET 'https://<bmc_ip>/redfish/
v1/UpdateService/FirmwareInventory/golden_image_nic'
```

7.8.1.4 Updating Golden Images Using Redfish

The process for updating golden images via Redfish includes the following steps.

1. The process for updating golden images via Redfish includes the following steps:
   - For NIC golden image:

     ```
     curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST -d
     '{"TransferProtocol":"HTTP", "ImageURI":"<remote-server-ip>/<nic-golden-image-path>", "Targets":
     ["redfish/v1/UpdateService/FirmwareInventory/golden_image_nic"]}' https://<bmc-ip>/redfish/v1/
     UpdateService/Actions/UpdateService.SimpleUpdate
     ```

   - For Arm golden image:

     ```
     curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST -d
     '{"TransferProtocol":"HTTP", "ImageURI":"<remote-server-ip>/<arm-golden-image-path>", "Targets":
     ["redfish/v1/UpdateService/FirmwareInventory/golden_image_arm"]}' https://<bmc-ip>/redfish/v1/
     UpdateService/Actions/UpdateService.SimpleUpdate
     ```

     Parameters:
     - `ImageURI` - Specify the image location in the format `<remote-server-ip>/<golden-image-path>`
     - `bmc-ip` - Specify the IP address of the BMC

       After initiating the update, a new task is created for monitoring progress, with a sample response:

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

2. Track the update's progress using the following command:

   ```
   curl -k -u root:'<password>' -X GET https://<bmc-ip>/redfish/v1/TaskService/Tasks/<task-id>
   ```

   Update states:
   - `0%` - Update started.
   - `10%` - Update in progress.
   - `100%` - Update complete.

     Expected output after a successful update:

     ```
     "PercentComplete": 100,
     "TaskState": "Completed",
     "TaskStatus": "OK"
     ```

     In case of failure, reboot the BMC and retry the update.

     > The golden image update process typically takes 1-3 minutes to complete.

If the candidate image version matches the one stored in the BMC's non-volatile memory, the update is skipped by default, and the following response is returned:

```
{
  ...
  {
```

```
      "@odata.type": "#MessageRegistry.v1_4_1.MessageRegistry",
      "Message": "The update operation for the component 'BMC' is skipped because 'Component image is identical'.",
      "MessageArgs": [
        "BMC",
        "Component image is identical"
      ],
      "MessageId": "NvidiaUpdate.1.0.ComponentUpdateSkipped",
      "Resolution": "Retry firmware update operation with the force flag",
      "Severity": "OK"
    },
  ],
  ...
  "TaskState": "Completed",
  "TaskStatus": "OK"
}
```

To override the default behavior and force the update, include the `"ForceUpdate": true` parameter in the command:

- For NIC golden image:

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST -d '{"TransferProtocol":"HTTP",
"ImageURI":"<remote-server-ip>/<nic-golden-image-path>", "Targets":["redfish/v1/UpdateService/
FirmwareInventory/golden_image_nic"], "ForceUpdate": true}' https://<bmc-ip>/redfish/v1/UpdateService/
Actions/UpdateService.SimpleUpdate
```

- For Arm golden image:

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST -d '{"TransferProtocol":"HTTP",
"ImageURI":"<remote-server-ip>/<arm-golden-image-path>", "Targets":["redfish/v1/UpdateService/
FirmwareInventory/golden_image_arm"], "ForceUpdate": true}' https://<bmc-ip>/redfish/v1/UpdateService/
Actions/UpdateService.SimpleUpdate
```

7.8.1.5 OOB Network Configuration

To enhance the system's security, a new mechanism has been introduced to control network connectivity over the OOB network. This new feature provides an IPMI command to disable any communication between the BlueField BMC, BlueField, and the OOB management network. A set of IPMI commands are introduced to selectively enable the network on each of the above interfaces. This permits the platform's RoT to have complete control over which network interfaces can be enabled and when.

This IPMI can only be sent by the platform's ROT. OOB and BlueField are blocked.

By default, the OOB interface is enabled. However, for the host BMC to gain control over this interface, it must disable it during the initial boot. Once disabled, the interface remains in that state regardless of BMC reboots or system cold boots.

For more details, refer to "OOB Network 3-Port Switch Control".

7.8.1.6 Golden Images Reprovisioning

The re-provisioning flow is initiated using the following IPMI command:

```
bmc> ipmitool raw 0x32 0x99 <golden_image_timeout> <timeout_from_network> <verbosity_level> <halt_hard_reset>
```

This command must be executed from within the BMC, as it can significantly impact the system. Upon execution:

1. The golden images are extracted from the BlueField BMC's non-volatile memory.
2. The recovery process is initiated, pushing the golden images to the RShim.

3. The RShim console output is redirected to the BMC console, allowing the user to monitor the process.

Once the process completes, both the BlueField NIC and ARM execute the designated golden images retrieved from a preconfigured server.

Command parameters:

- `<golden_image_timeout>` - Timeout value (in minutes) for updating the golden images. Default Value: `15` minutes (input `0` to use the default).
- `<timeout_from_network>` - Timeout value (in minutes) for booting the operational image from the network. Default Value: `60` minutes (input `0` to use the default).
- `<verbosity_level>` - Controls the level of detail displayed during the reprovisioning process:
  - `0` - Quiet mode (only error messages are displayed)
  - `1` - Info mode (error messages and reprovisioning process messages are displayed)
  - `2` - Full mode (all messages, including BlueField RShim messages, are displayed)
- `<halt_hard_reset>` (Optional) - Specifies whether to halt the reprovisioning process before performing the final hard reset of the BlueField.

  > ⚠ The final hard reset is crucial to activate the NIC firmware installed from the network.

  - `0` - Perform the hard reset to complete the reprovisioning process (default)
  - `1` - Halt the process before the final hard reset

  > ℹ Reprovisioning messages are prefixed with `[<running date> GOLDEN-IMAGE-RECOVERY]`.

7.8.1.7 Signaling BlueField BMC After Arm OS Programming Completion

After BFB installation is complete, the BlueField BMC waits for a specific sequence of messages over the RShim log:

```
NIC firmware update done    # Indicates that the firmware update for the NIC subsystem has been successfully
completed
Installation finished       # Signals the completion of the installation process for the BFB from the network
Linux up                    # Indicates that the BlueField BMC has acknowledged that the Arm OS has booted up and
is ready
```

BlueField BMC expects these messages in the specified order.

7.8.1.8 Adding Entries to RShim Log from BlueField Arm OS

Users can add custom entries to the RShim log from the BlueField Arm OS using the `bfrshlog` command. The syntax of the command is: `bfrshlog <output>`.

For example, to add the message "Linux up" to the RShim log, run:

```
bfrshlog "Linux up"
```

7.8.1.9 Expected Output

- All output from the BlueField Arm console is redirected to the BlueField BMC console for monitoring purposes.
- The steps of the re-provisioning process are printed with `[<running date> GOLDEN-IMAGE-RECOVERY]` prefix and are outlined in the following:

```
[<running date> GOLDEN-IMAGE-RECOVERY]  Checking pcie slot is in reset
[<running date> GOLDEN-IMAGE-RECOVERY]  Read golden images from flash
[<running date> GOLDEN-IMAGE-RECOVERY] Set FNP to 0
[<running date> GOLDEN-IMAGE-RECOVERY] Checking rshim interface after SOC hard reset
[<running date> GOLDEN-IMAGE-RECOVERY]  Starting ATF/UEFI golden image update
[<running date> GOLDEN-IMAGE-RECOVERY] Finished updating ATF/UEFI golden image
[<running date> GOLDEN-IMAGE-RECOVERY] Starting NIC FW golden image update
[<running date> GOLDEN-IMAGE-RECOVERY] Finished updating NIC FW golden image
[<running date> GOLDEN-IMAGE-RECOVERY] Stop Redfish server
[<running date> GOLDEN-IMAGE-RECOVERY] Configure Recovery image to boot from network
[<running date> GOLDEN-IMAGE-RECOVERY] set FNP to 1
[<running date> GOLDEN-IMAGE-RECOVERY] Booting BFB from network
[<running date> GOLDEN-IMAGE-RECOVERY] Start Redfish server
[<running date> GOLDEN-IMAGE-RECOVERY] Set boot option to default
if halt_hard_reset is 0:
[<running date> GOLDEN-IMAGE-RECOVERY] Finished programming image from network. Start DPU hard reset
if halt_hard_reset is 1:
[<running date> GOLDEN-IMAGE-RECOVERY] Finished programming image from network
[<running date> GOLDEN-IMAGE-RECOVERY] The Reprovisioning process was halted at user's request. To complete the
process, please power cycle the device
```

A failed update prints the following:

```
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: aborting process! PCIE is not in reset.
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: Reading golden_image_nic failed
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: Reading golden_image_arm failed
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: rshim has not started successfully
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: pushing ATF/UEFI golden image over rshim failed
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: programming of ATF/UEFI golden image failed
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: pushing NIC FW golden image over rshim failed
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: programming of NIC FW golden image failed
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: failed to configure image to boot from network
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR:  programming of image from network failed: NIC firmware update failed
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: programming of image from network failed: Installation failed
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: programming of image from network failed: Failed to get Linux up
```

Due to line buffering in the BlueField Arm console, buffered output lines receive the same timestamp value in `<running date>` when they are redirected to the BlueField BMC console.

7.8.2 Power Capping

> 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 BlueField reboot.

7.8.2.1 Redfish Power Capping Requests

7.8.2.1.1 Getting General Power Capping Information

Control information:

```
curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/Chassis/Card1/Controls/PowerLimit_0
```

Output example:

```
{
  "@odata.id": "/redfish/v1/Chassis/Card1/Controls/PowerLimit_0",
  "@odata.type": "#Control.v1_3_0.Control",
  "AllowableMax": 300,
  "AllowableMin": 200,
  "ControlMode": "Manual",
  "ControlType": "Power",
  "Id": "PowerLimit_0",
  "Name": "System Power Control",
  "SetPoint": 10,
  "SetPointUnits": "%",
  "Status": {
    "Health": "OK",
    "HealthRollup": "OK",
    "State": "Enabled"
  }
}
```

7.8.2.1.2 Enabling/Disabling Adjustment of Power Capping

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X PATCH https://<bmc_ip>/redfish/v1/Chassis/
Card1/Controls/PowerLimit_0 -d '{"ControlMode":<"Manual"/"Disabled">}'
```

The ability to adjust power capping is disabled by default.

7.8.2.1.3 Setting Power Allocation Percentage

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X PATCH https://<bmc_ip>/redfish/v1/Chassis/
Card1/Controls/PowerLimit_0 -d '{"SetPoint": <val>}'
```

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

If user configuration is lower than the minimum capacity power or higher than the maximum capacity power, then BMC will return error.

7.8.2.2 IPMI Power Capping Commands

7.8.2.2.1 Getting Power Capping Status

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

7.8.2.2.2 Enabling/Disabling Adjustment of Power Capping

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

Where `val` :

- 0 - disable
- 1 - enable

Permissions

Changeable only from BMC prompt using `admin` account.

The ability to adjust power capping is disabled by default.

7.8.2.2.3 Getting Power Capping Percentage

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

7.8.2.2.4 Setting Power Capping Percentage

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

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

Permissions

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
```

7.8.2.2.5 Getting Maximum Power Capacity

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

Power is given in watts.

7.8.2.2.6 Getting Minimum Power Capacity

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

Power is given in watts.

7.8.3 RShim Over USB

7.8.3.1 Network Connection from BMC to BlueField

By default, the BMC and NVIDIA® 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

7.8.3.2 Enabling RShim on BlueField BMC

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

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

> 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
```

> RShim can be force enabled on BMC even it is started on host. The previous steps are not necessary in this case:
>
> ```
> curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST -d '{"Rshim": "Forced"}'
> https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Actions/Oem/NvidiaManager.SetRshim
> ```
>
> > After force enable is performed, the RShim will be in drop mode if it is enabled on host. The host can regain RShim using `systemctl restart rshim` after the BMC disables RShim.

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

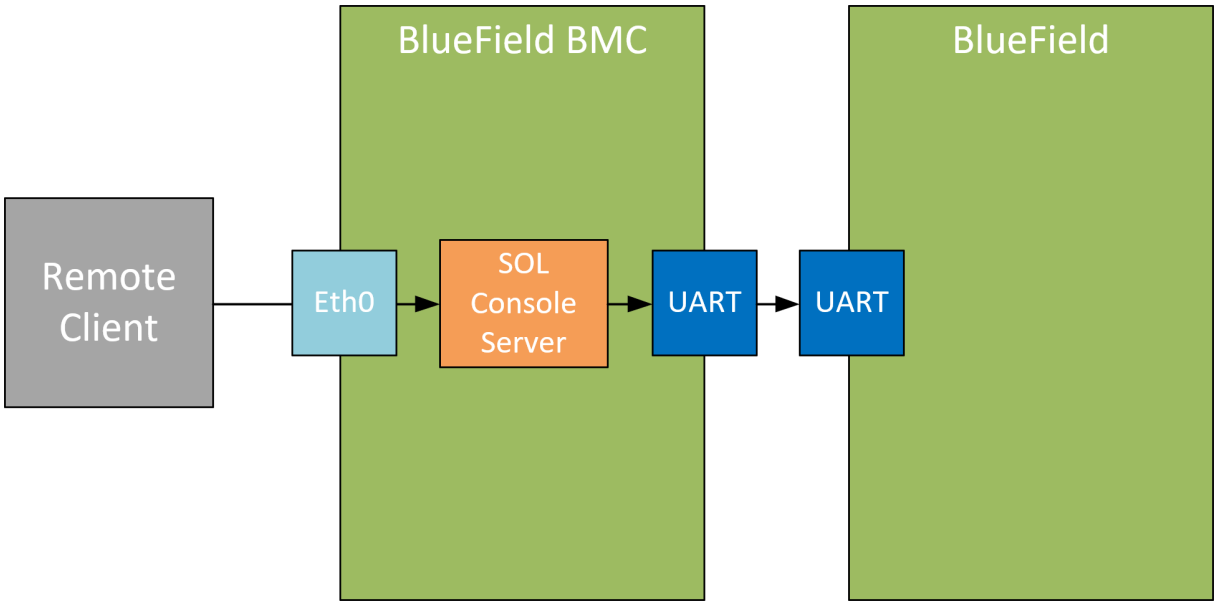
```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X GET https://<bmc_ip>/redfish/v1/
Managers/Bluefield_BMC/Oem/Nvidia
```

> 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.

7.8.4 Serial Over LAN

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.

> ⚠ The serial-over-LAN (SOL) connection is first established using the BlueField BMC credentials. Once the connection to the BlueField BMC is successful, the serial communication is redirected to the BlueField Arm console, where additional BlueField Arm credentials are required to complete the connection.



7.8.4.1 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": truethe
```

```
    }
  },
...
}
```

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.

7.8.4.2 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

No.	Function	Command	Description
4	Deactivate SOL	`ipmitool -C 17 -I lanplus -U <username> -P <password> -H <ip_address> sol deactivate`	Deactivate SOL access to the BlueField console

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

7.8.4.3 SysRq Support in SOL

SysRq is a special key combination used by Linux to perform various low-level commands. SOL invokes the SysRq feature by sending a serial break signal, followed by the desired key. To enable SysRq, the user must issue the following command on the BlueField:

```
echo 1 > /proc/sys/kernel/sysrq
```

In the SOL of BMC, the break signal is generated by keys `\n~B`. So, in an SOL session, the user may enter the `\n~B` keys to trigger the break and then enter a keycode as the SysRq command.

For example, the following are the outputs when inputting `h` after generating a break signal in the SOL session:

- For SOL over IPMI:

```
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:
bluefield login: ~B [send break]
[490472.371785] sysrq: HELP : loglevel(0-9) reboot(b) crash(c) terminate-all-tasks(e) memory-full-oom-
kill(f) kill-all-tasks(i) thaw-filesystems(j) sak(k) show-backtrace-all-active-cpus(l) show-memory-usage(m)
nice-all-RT-tasks(n) poweroff(o) show-registers(p) show-all-timers(q) unraw(r) sync(s) show-task-states(t)
unmount(u) show-blocked-tasks(w) dump-ftrace-buffer(z)
```

- For SOL over SSH (break signal generated with `\n~~B`):

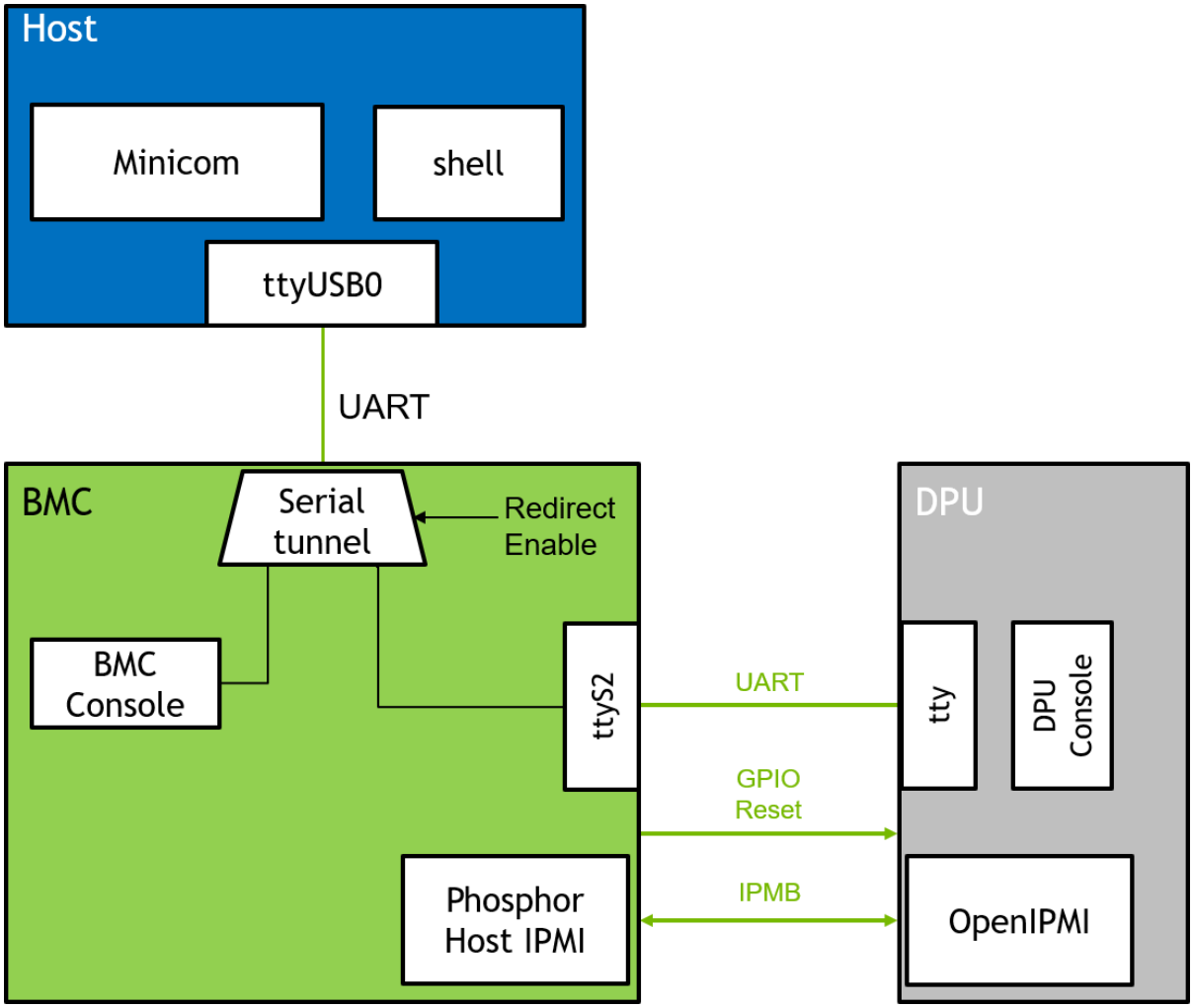
```
ssh -p 2200 root@10.10.10.10
root@10.10.10.10's password:

bluefield login:
[490472.371785] sysrq: HELP : loglevel(0-9) reboot(b) crash(c) terminate-all-tasks(e) memory-full-oom-
kill(f) kill-all-tasks(i) thaw-filesystems(j) sak(k) show-backtrace-all-active-cpus(l) show-memory-usage(m)
nice-all-RT-tasks(n) poweroff(o) show-registers(p) show-all-timers(q) unraw(r) sync(s) show-task-states(t)
unmount(u) show-blocked-tasks(w) dump-ftrace-buffer(z)
```

⚠ In the context of SOL over SSH connections, an additional tilde symbol ('`~`') is used to navigate through multiple layers of SSH sessions to access the SOL session (e.g., `\n~~B`).

7.8.5 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.

The following sections list the supported commands.

7.8.5.1 Enabling Serial Redirect Mode to Run from Arm or BMC OS

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

1. Disables vendor field mode if enabled.

2. Enables tunneling on BMC through UART by default.
3. BlueField BMC validates that BlueField is in controller mode (refer to the self-hosted SKUs), and if so, it resets (`SOC_HARD_RESET`) the BlueField.

```
ipmitool raw 0x32 0x6D 0x01
```

7.8.5.2 Disabling Serial Redirect Mode Settings from Being Run on Arm or BMC OS

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.

```
ipmitool raw 0x32 0x6D 0x00
```

7.8.5.3 Fetching Serial Redirect Mode Settings

```
ipmitool raw 0x32 0x6E
```

7.8.5.4 Starting Tunneling on BMC Through UART

Run the following command on the host where BMC is connected:

```
/usr/bin/nvidia-field-mode-modifier starttunnel
```

7.8.5.5 Stopping 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.

User can also stop tunneling by running the command `~.` on the console client.

> If on an SSH connection, 'escape' the `~` character by entering it twice: `~~.`

7.8.6 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.

7.8.6.1 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
```

7.8.6.2 Updating CEC Firmware with Vendor Field Mode

⚠ 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.

7.8.6.3 Updating BMC and Glacier Firmware with Vendor Field Mode

⚠ 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
```

7.8.6.4 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`

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

Operation Description	Command
To check the CEC firmware update status on BMC ⚠ 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.
Transfer CEC firmware image for firmware update through UART ⚠ 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.

7.9 Reset Control

⚠ Rebooting NVIDIA® BlueField®-2 immediately after rebooting its BMC is restricted. The user should wait until the IPMI service becomes operational before rebooting BlueField-2, with a recommended wait of 30 seconds.

7.9.1 Reset Control Using Redfish

Issue the following command from the BMC to get the power status of the BlueField networking platform (DPU or SuperNIC):

```
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",
 …
}
```

7.9.1.1 Hard Reset of BlueField Arm Cores and NIC Subsystem

A hard reset of BlueField is permitted only when all connected hosts assert the PERST signal. It is crucial for all host devices to assert the PERST signal when BlueFiled-3 is shared among multiple hosts to enable a hard reset.

```
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"}'
```

Refer to the "RedFish Post (Action)" section in Redfish Specification DSP0266 for an example of successful action response.

7.9.1.2 Force Hard Reset of BlueField Arm Cores and NIC Subsystem

Force hard reset of the BlueField happens without waiting for `All_STANDBY` or `PERST`. Users must make sure the server is ready for the reset!

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST https://<bmc_ip>/redfish/v1/Systems/
Bluefield/Oem/Nvidia/SOC.ForceReset
```

Refer to the "RedFish Post (Action)" section in Redfish Specification DSP0266 for an example of successful action response.

7.9.1.3 Hard Reset of BlueField Arm Cores

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

Refer to the "Redfish Post (Action)" section in Redfish Specification DSP0266 for an example of successful action response.

7.9.1.4 Soft Shutdown of BlueField Arm OS

This command is relevant only for BlueField-3 devices.

The following is supported when operating in DPU mode only.

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

Refer to the "RedFish Post (Action)" section in Redfish Specification DSP0266 for an example of successful action response.

7.9.1.4.1 Monitoring BlueField Arm OS Shutdown with Redfish

When the BlueField Arm OS shuts down successfully, `PowerState` changes to `Paused` and `StatusState` changes to `StandbyOffline`.

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

Example output:

```
…
"PowerState": "Paused",
…
 "Status": {
    "Health": "OK",
    "HealthRollup": "OK",
    "State": "StandbyOffline"
  },
…
```

7.9.2 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 BlueField:

```
ipmitool chassis power status
```

To perform a reset operation on BlueField, use the following IPMI commands:

Description	Command
Hard reset of BlueField (Arm cores and NIC)	`ipmitool chassis power cycle`
Hard reset of BlueField Arm cores	`ipmitool chassis power reset`
Soft Shutdown of BlueField Arm OS ⚠ This command is relevant only for BlueField-3.	`ipmitool power soft`

These commands update the most recent restart cause, which can be retrieved using `ipmitool chassis restart_cause`. The value will be reported as "Chassis Control Command".

⚠ A hard reset of BlueField is permitted only when all connected hosts assert the PERST signal. This is particularly important when BlueField-3 is shared among multiple hosts. In such cases, each host must assert PERST to ensure that a hard reset can be safely executed.

⚠ Soft shutdown of BlueField Arm OS is allowed only when the Arm OS is running. To retrieve the Arm OS state, refer to the `0xA3` command under "IPMItool NIC Subsystem Management".

⚠ Between each reset control, there should be a wait until the system finishes the operation.
- 20-second wait in BlueField-2
- 5-second wait in 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 the BlueField:

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 ℹ This reset command is only available when the BlueField Arm OS is up. This option updates the latest restart cause, retrieved via `ipmitool chassis restart_cause`, to "Soft Reset". • `0x03` - reset on-board 3-port switch

7.9.2.1 Monitoring BlueField OS Shutdown Using IPMI

After a successful shutdown, the BlueField Arm enters a low-power standby state.

> The BlueField Arm cannot be fully powered off, and Standby is its final state

To get the BlueField's OS state, refer to the `0xA3` command under "IPMItool NIC Subsystem Management".

To get the BlueField Arm to boot back to the BlueField Arm OS, users can either power cycle BlueField or perform a hard reset of the BlueField Arm.

> The output of IPMItool chassis power status will show "Chassis power is on".

7.10 Factory Reset

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

7.10.1 Step 1 – Reset BlueField BMC to Factory Default

1. Run the following command:

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST https://<BF-BMC-IP>/redfish/v1/
Managers/Bluefield_BMC/Actions/Manager.ResetToDefaults -d '{"ResetToDefaultsType": "ResetAll"}'
{
  "@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The request completed successfully.",
      "MessageArgs": [],
      "MessageId": "Base.1.15.0.Success",
      "MessageSeverity": "OK",
      "Resolution": "None"
    }
  ]
}
```

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

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

7.10.2 Step 2 – Wipe BlueField eMMC and SSD Storage

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

- eMMC:

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

- NVMe:

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

7.10.3 Step 3 – Reset UEFI to Factory Default

Use the Redfish BIOS Settings PATCH command:

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

7.10.4 Step 4 – Reset NIC TLVs to Factory Default

Run the following from the Arm console:

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

7.11 DPU BMC SPDM Attestation via Redfish

The DPU BMC attestation process enables secure verification of device identity and firmware integrity using standardized protocols. This implementation leverages SPDM (Security Protocol and Data Models) over MCTP (Management Component Transport Protocol) to provide remote attestation capabilities via the Redfish API.

7.11.1 Redfish Commands

> ⚠ For detailed information about the DPU attestation process, measurement descriptions, and reference values, refer to the DPU Attestation documentation.

7.11.1.1 Get ComponentIntegrity Collection

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

This command returns a collection of all attestation targets in the system.

In DPU BMC, the available attestation targets are:

- `Bluefield_DPU_IRoT` - The BlueField IRoT (Initial Root of Trust), a Platform Security Controller (PSC) that stores measurements related to the Arm and NIC components
- `Bluefield_ERoT` - The BlueField BMC ERoT (External Root of Trust), which contains measurements related to the DPU BMC

7.11.1.2 Get Certificate Chain of Specific Attestation Target

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X GET https://<bmc ip>/redfish/v1/Chassis/
<target-id>/Certificates/CertChain
```

This command retrieves the certificate chain for a specific attestation target. The response is a JSON structure containing the entire certificate chain, which can be used to verify the authenticity of the component.

7.11.1.3 Get Measurements from Attestation Target

```
# 1. Request all available measurements
curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST \
  https://<bmc ip>/redfish/v1/ComponentIntegrity/<target id>/Actions/ComponentIntegrity.SPDMGetSignedMeasurements

# 2. Request specific measurements
curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST \
  -d '{"SlotId": 0, "MeasurementIndices": [2,5], "Nonce":
"d42a0594c5cd5743ee08fe5ec3cf884b1fac4f106879cda98b7d1c51652b04b7"}' \
  https://<bmc ip>/redfish/v1/ComponentIntegrity/HGX_IRoT_NIC_0/Actions/
ComponentIntegrity.SPDMGetSignedMeasurements
```

This command retrieves signed measurements from the specified component.

Parameters:

1. Nonce
    - Description: A unique, randomly generated value used to prevent replay attacks.
    - Format: 32-byte (64-character) hexadecimal string.
    - Usage:
        - Must be generated and provided by the client for each request.
        - Ensures that each request is fresh and secure.
2. Certificate Slot ID
    - Description: Indicates which slot contains the certificate chain used for signing.
    - Supported Value: `0`
    - Default: `0`
    - Note: Only Slot 0 is supported, which holds the NVIDIA certificate chain.
3. Measurement Indices
    - Description: Specifies the measurement indices to request.
    - Format: Array of integers.
    - Default: If omitted, `0xFF` is used to request all available measurements.

7.11.1.3.1 Handling the Response

This operation is asynchronous and returns a task object rather than the measurement data itself.

Example response:

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

7.11.1.3.2 Monitoring Task Progress

Periodically check the task until completion using:

```
curl -k -u root:'<password>' -H "Content-Type: application/json" \
  -X GET https://<bmc ip>/redfish/v1/TaskService/Tasks/<id>
```

A completed task appears as

```
{
  ...
  "PercentComplete": 100,
  ...
  "TaskState": "Completed",
  "TaskStatus": "OK"
}
```

7.11.1.4 Get Measurements Response Data

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X GET \
  https://<bmc ip>/redfish/v1/ComponentIntegrity/<target id>/Actions/ComponentIntegrity.SPDMGetSignedMeasurements/
data
```

This command retrieves the signed measurement data previously requested via the `SPDMGetSignedMeasurements` action.

Example output:

```
{
  "HashingAlgorithm": "TPM_ALG_SHA_512",
  "SignedMeasurements": "<base64 encoded measurements>",
  "SigningAlgorithm": "TPM_ALG_ECDSA_ECC_NIST_P384",
  "Version": "1.1.0"
}
```

7.11.2 Redfish Event Log

Each time a new *Get Measurements* command is issued, a Redfish event log entry is generated.

Example entry:

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<id>",
  "@odata.type": "#LogEntry.v1_15_0.LogEntry",
  "Created": "<date>",
  "EntryType": "Event",
  "Id": "<id>",
  "Message": "Redfish attestation measurements POST request received",
  "Modified": "<date>",
  "Name": "System Event Log Entry",
  "Resolved": false,
  "Severity": "OK"
}
```

8 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.

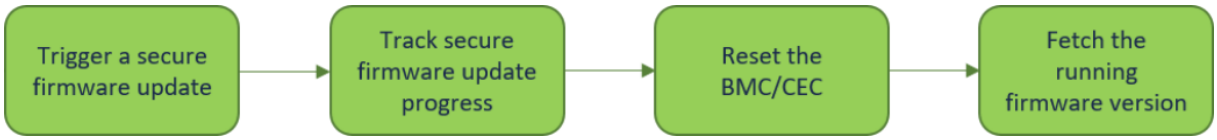
The primary software components of BMC are the following:

- U-boot bootloader
- Linux kernel
- OpenBMC distro

This section consists of the following pages:

8.1 CEC and BMC Firmware Operations

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



8.1.1 CEC and BMC Firmware Commands

8.1.1.1 Triggering Secure Firmware Update

> Required for BMC/CEC update.

This section describes how to trigger a secure update and to track the secure update progress.

- To perform a multipart update with `MultipartHttpPushUri` API, run:

```
curl -k -u root:'<password>' https://<bmc_ip>/redfish/v1/UpdateService/update-multipart -F
'UpdateParameters={};type=application/octet-stream' -F UpdateFile=@<package_path>
```

- Update with `HttpPushUri` API - Deprecated

`HttpPushUri` API is obsolete. NVIDIA recommends users migrate to `MultipartHttpPushUri` API as support for `HttpPushUri` API will be discontinued in the future.

```
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

8.1.1.2 Tracking Secure Firmware Update Progress

Required for BMC/CEC update.

```
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 by running:

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

Where:

- `password` - password of root user
- `bmc_ip` - BMC IP address
- `task_id` - Task ID

To retrieve additional information about the task progress, run:

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

The task status can be one of the following:

- `"Running"` - the update is currently in progress.
- `"Completed"` - the update finished successfully.
- `"Exception"` - an error occurred during the update.

8.1.1.3 Resetting/Rebooting BMC

Required for BMC update.

To reset/reboot BMC, run:

```
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

8.1.1.4 Getting the Running BMC Firmware Version

> Required for BMC update.

The following commands get the running firmware version of BlueField devices via the BMC.

- For NVIDIA® 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

- For NVIDIA® BlueField®-2:

  ```
  curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory
  ```

  Get 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.

8.1.1.5 Getting the Running CEC Firmware Version

> Required for CEC update.

Gets the running firmware version from CEC.

```
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

8.1.1.6 ForceUpdate

> Relevant for BlueField-3 only.

> Required for BMC update.

`ForceUpdate` forces the update procedure to proceed even if the target version is identical to the currently programmed version.

- When using `HttpPushUri` API, `ForceUpdate` is always `true` (i.e., forces the update procedure to proceed even if the target version is identical)
- When using `MultipartHttpPushUri` API, `ForceUpdate` is `false` (i.e., the update procedure fails if the target version is identical), unless `"ForceUpdate":true` is included under `UpdateParameters`:

  ```
  curl -k -u root:'<password>' https://<bmc_ip>/redfish/v1/UpdateService/update-multipart -F
  'UpdateParameters={"ForceUpdate":true};type=application/octet-stream' -F UpdateFile=@<package_path>
  ```

  Where:

  - `password` - password of root user
  - `bmc_ip` - BMC IP address

8.1.2 Updating BMC

> Firmware update takes about 12 minutes.

After initiating the BMC secure update, you can expect to receive responses similar to the following.

- If an `HttpPushUri` API is used:

```
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": "<id>",
  "TaskState": "Running"
}
```

- If a `MultipartHttpPushUri` API us used:

```
curl -k -u root:'<password>' https://<bmc_ip>/redfish/v1/UpdateService/update-multipart -F
'UpdateParameters={};type=application/octet-stream' -F UpdateFile=@<package_path>
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/0",
  "@odata.type": "#Task.v1_4_3.Task",
  "Id": "<id>",
  "TaskState": "Running",
  "TaskStatus": "OK"
}
```

Where:

- `package_path` - the BMC firmware image file including its path

For example:

```
curl -k -u root:'myP@ssword_12345!' https://10.10.10.10/redfish/v1/UpdateService/update-multipart
-F 'UpdateParameters={};type=application/octet-stream' -F UpdateFile=@/root/bf2-bmc-ota-24.01-4-
ipn.tar
```

The following command is used to track secure firmware update progress:

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

The task is completed when `PercentComplete` reaches 100.

Since the reboot option is disabled during the update procedure, use the following command to reboot the BMC.

```
curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/TaskService/Tasks/<id> | 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"
    }
  ]
}
```

The following commands are 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:

  a. Get 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"
  …
  ```

  b. Use the following command with the retrieved firmware ID from the previous step.

  ```
  curl -k -u root:'<password>' -X GET https:/<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory/
  8c8549f3_BMC_Firmware | 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
  ```

For BlueField-3 BMC only: When updating firmware to a target that has an identical version using `MultipartHttpPushUri`, you must include `ForceUpdate=true`. Otherwise, the update procedure fails with the message `Component image is identical`.

```
curl -k -u root:'<password>' https://<bmc_ip>/redfish/v1/UpdateService/update-multipart -F
'UpdateParameters={"ForceUpdate":true};type=application/octet-stream' -F UpdateFile=@<package_path>
```

8.1.3 Updating CEC

> Firmware update takes about 20 seconds.

After initiating the BMC secure update, expect responses similar to the following--depending whether `HttpPushUri` or `MultipartHttpPushUri` is used:

- `HttpPushUri` API:

  ```
  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"
  ```

- `MultipartHttpPushUri` API:

  ```
  curl -k -u root:'<password>' https://<bmc_ip>/redfish/v1/UpdateService/update-multipart -F
  'UpdateParameters={};type=application/octet-stream' -F UpdateFile=@<package_path>
  {
    "@odata.id": "/redfish/v1/TaskService/Tasks/0",
    "@odata.type": "#Task.v1_4_3.Task",
    "Id": "0",
    "TaskState": "Running",
    "TaskStatus": "OK"
  }
  ```

  Where:

  - `package_path` - the BMC firmware image file including its path
    For example:

    ```
    curl -k -u root:'myP@ssword_12345!' https://10.10.10.10/redfish/v1/UpdateService/update-multipart
    -F 'UpdateParameters={};type=application/octet-stream' -F UpdateFile=@/root/
    cec_ota_BMGP-04.0f_debug.bin
    ```

Use the following command to track the progress of the CEC firmware update.

```
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, CEC activation and reset should be done (see "Activating New CEC") to apply the changes once the update is finished.

Use the following command 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
```

8.1.4 Activating New CEC

> ⚠ This is relevant only for BlueField-3 networking platforms (DPU or SuperNIC).

To activate the new CEC firmware, you must reset the CEC device. The following sub-sections describe possible reset options.

8.1.4.1 Resetting CEC and BMC Subsystems Using CEC Self-reset Command

> ⚠ The CEC self-reset command is supported in CEC versions `00.02.0180.0000` and above. The command activates the new firmware on CEC and should only be used after the CEC update procedure is complete.

8.1.4.1.1 Redfish Command

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

Where:

- `password` - password of root user
- `bmc_ip` - BMC IP address

Command response options:

- The response in case of success:

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

- The response if there is no pending CEC firmware:

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST https://<bmc_ip>/redfish/v1/
Chassis/Bluefield_ERoT/Actions/Chassis.Reset -d '{"ResetType":"GracefulRestart"}'
{
  "error": {
    "@Message.ExtendedInfo": [
      {
        "@odata.type": "#Message.v1_1_1.Message",
        "Message": "The requested resource of type ERoT FW named 'Pending-ERoT-FW' was not found.",
        "MessageArgs": [
          "ERoT FW",
          "Pending-ERoT-FW"
```

```
        ],
        "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 ERoT FW named 'Pending-ERoT-FW' was not found."
  }
}
```

- The response if the command is not supported by the current CEC firmware:

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST https://<bmc_ip>/redfish/v1/
Chassis/Bluefield_ERoT/Actions/Chassis.Reset -d '{"ResetType":"GracefulRestart"}'
{
  "error": {
    "@Message.ExtendedInfo": [
      {
        "@odata.type": "#Message.v1_1_1.Message",
        "Message": "The action ERoT self-reset is not supported by the resource.",
        "MessageArgs": [
          "ERoT self-reset"
        ],
        "MessageId": "Base.1.15.0.ActionNotSupported",
        "MessageSeverity": "Critical",
        "Resolution": "The action supplied cannot be resubmitted to the implementation.  Perhaps the action
was invalid, the wrong resource was the target or the implementation documentation may be of assistance."
      }
    ],
    "code": "Base.1.15.0.ActionNotSupported",
    "message": "The action ERoT self-reset is not supported by the resource."
  }
}
```

8.1.4.1.2 IPMI Command

```
ipmitool raw 0x32 0xD2
```

Command response options:

- If successful, no response is given. Glacier and BMC are reset.
- The response if there is no pending CEC firmware is:

```
Unable to send RAW command (channel=0x0 netfn=0x32 lun=0x0 cmd=0xd2 rsp=0xd6): Cannot execute command,
command disabled
```

- The response if the command is not supported by the current CEC firmware is:

```
Unable to send RAW command (channel=0x0 netfn=0x32 lun=0x0 cmd=0xd2 rsp=0xd5): Command not supported in
present state
```

8.1.4.1.3 Log Event Entries Created per Response Type

- Log event entry created in case of success:

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>/attachment",
  "Created": "<Date>",
  "EntryType": "Event",
  "Id": "<Id>",
  "Message": "The request completed successfully.",
  "MessageArgs": [
    ""
  ],
  "MessageId": "Base.1.0.Success",
  "Name": "System Event Log Entry",
  "Resolution": "Ready for ERoT self Reset",
  "Resolved": false,
  "Severity": "OK"
}
```

- Log event entry created if there is no pending CEC firmware:

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X GET https://<bmc_ip>/redfish/v1/
Systems/Bluefield/LogServices/EventLog/Entries/11
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>/attachment",
  "Created": "<Date>",
  "EntryType": "Event",
  "Id": "<Id>",
  "Message": "Awaiting for an action to proceed with installing image '' on 'ERoT'.",
  "MessageArgs": [
    "",
    "ERoT"
  ],
  "MessageId": "Update.1.0.AwaitToUpdate",
  "Name": "System Event Log Entry",
  "Resolution": "Cannot perform ERoT self reset: There is no EC FW pending. Perform an ERoT FW update.",
  "Resolved": false,
  "Severity": "OK"
}
```

- Log event entry created if the command is not supported by the current CEC firmware:

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>/attachment",
  "Created": "<Date>",
  "EntryType": "Event",
  "Id": "<Id>",
  "Message": "The action ERoT self Reset is not supported by the resource.",
  "MessageArgs": [
    "ERoT self Reset"
  ],
  "MessageId": "Base.1.0.ActionNotSupported",
  "Name": "System Event Log Entry",
  "Resolution": "Cannot perform ERoT self reset: The action is not supported by the current ERoT version.",
  "Resolved": false,
  "Severity": "OK"
}
```

- Possible log event entries created if an error occurs during the BMC shutdown procedure (received after a success log event entry):

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>",
...
  "MessageId": "Base.1.0.InternalError",
  "Name": "System Event Log Entry",
  "Resolution": "ERoT Reset: Failed to close services towards ERoT reset",
  "Resolved": false,
  "Severity": "Critical"
}
```

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>",
...
  "MessageId": "Base.1.0.InternalError",
  "Name": "System Event Log Entry",
  "Resolution": "ERoT Reset: Isolate operation may still be in progress.",
  "Resolved": false,
  "Severity": "Critical"
}
```

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>",
...
  "MessageId": "Base.1.0.InternalError",
  "Name": "System Event Log Entry",
  "Resolution": "ERoT Reset: Failed to unmount file system towards ERoT reset.",
  "Resolved": false,
  "Severity": "Critical"
}
```

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>",
...
  "MessageId": "Base.1.0.InternalError",
  "Name": "System Event Log Entry",
  "Resolution": ERoT Reset: Failed to unmount file system. It is still mounted as read-write (rw),
  "Resolved": false,
  "Severity": "Critical"
}
```

8.1.4.2 Resetting CEC and BMC Subsystems Using IPMI I2C Command over SMBus Channel Connected to PCIe Golden Finger

⚠ This option is valid only for servers which support I^2C over SMBus from the host BMC.

```
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
```

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

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

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

8.1.4.3 Resetting the Entire BlueField

This option typically involves a full power cycle of the host platform.

8.1.5 CEC Background Update Status

ℹ 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 devices. The firmware update is performed on the inactive copy. Upon a successful boot from the newly updated and active image, the inactive image (e.g., the previously active image) is updated with the latest image.

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

To check the status of the background update, run:

```
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
    }
  }
…
```

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

8.1.6 Possible Error Codes

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 after 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 that was 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 after 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` • A resolution is proposed: "Another update is in progress. Retry the update operation once it is complete." Check the status of the ongoing firmware update by looking at the `TaskCollection` resource.
Redfish task hangs	• Redfish task URI that was previously returned by the Redfish server is no longer accessible • PLDM-based firmware update progresses • After a reboot of BMC, or after 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.

Fault	Diagnosis and Possible Solution
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` • `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.1.15.0.PayloadTooLarge` and the `Resolution` "Firmware package size is greater than allowed size". Make sure the package size is less than the `UpdateService.MaxImageSizeBytes` property and retry the firmware update operation.

8.2 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 ...`

Scenario	Message
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`.

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

8.3 User Management

8.3.1 User Management Redfish Commands

8.3.1.1 Getting General Information

To retrieve 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
}
```

8.3.1.2 Listing Supported User Roles

To 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"
}
```

8.3.1.3 Listing 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"
}
```

8.3.1.4 Creating New User

To 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."
    }
  ]
}
```

8.3.1.5 Deleting User

To delete a user from the system:

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

Example:

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

8.3.2 User Management IPMI Commands

8.3.2.1 Listing Users

```
ipmitool user list [<channel-number>]
```

Example:

```
ipmitool user list 1
```

8.3.2.2 Creating User

```
ipmitool user set name <user-id> <username>
```

For example:

```
ipmitool user set name 2 Admin
```

8.3.2.3 Setting User Password

```
ipmitool user set password <user-id> <password>
```

Example:

```
ipmitool user set password 2 AdminPass_123
```

8.3.2.4 Enabling/Disabling User

```
ipmitool user <enable|disable> <user-id>
```

Example:

```
ipmitool user enable 2
```

8.3.2.5 Setting 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

Example:

```
ipmitool user priv 2 0x3 1
```

8.3.2.6 Enabling Remote IPMI for User

To enable remote IPMI command functionality for a user:

```
ipmitool channel setaccess [<channel-number>] <user-id> ipmi=<on|off>
```

For example:

```
ipmitool channel setaccess 1 2 ipmi=on
```

8.3.2.7 Lanplus Commands to Execute IPMI Commands Remotely for Admin Users

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
```

8.3.2.8 Lanplus Commands to Execute IPMI Commands Remotely for Non-admin Users

Lanplus commands to execute IPMI commands remotely for users with a non-administrator role:

```
ipmitool -C 17 -I lanplus -U <user> -P <password> -H <bmc-ip-address> -L <privilege (operator|user)> <ipmi-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
```

8.3.2.9 Deleting User

```
ipmitool user set name <user-id> ""
```

For example:

```
ipmitool user set name 2 ""
```

8.4 LLDP in Redfish

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

The LLDP schema provides the ability to enable/disable the LLDP in BMC and to get LLDP information from the BMC and from peer devices.

8.4.1 Getting LLDP Information

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET https://<bmc_ip>/redfish/v1/Managers/
Bluefield_BMC/DedicatedNetworkPorts/1
```

Output example:

```
{
    "@odata.id": "/redfish/v1/Managers/Bluefield_BMC/DedicatedNetworkPorts/1",
    "@odata.type": "#Port.v1_7_0.Port",
    "Ethernet": {
        "LLDPEnabled": true,
        "LLDPReceive": {
            "ChassisId": "MAC: 8c:85:c1:a2:ae:80",
            "ChassisIdSubtype": 4,
            "ManagementAddressIPv4": "10.60.7.238",
            "ManagementVlanId": 95,
            "PortId": "Ifname: 1/1/5",
            "PortIdSubtype": 5,
            "SystemCapabilities": [
                "Bridge",
                "Router"
            ],
            "SystemDescription": "Aruba JL676A  PL.10.13.0005",
            "SystemName": "MTL-T-F0-LAB-ORMANCE-SW-7-238"
        },
        "LLDPTransmit": {
            "ChassisId": "MAC: 94:6d:ae:5c:9d:cd",
            "ChassisIdSubtype": 4,
            "ManagementVlanId": 4095,
            "PortId": "MAC: 94:6d:ae:5c:9d:cd",
            "PortIdSubtype": 3,
            "SystemCapabilities": [
                "Station"
            ],
            "SystemDescription": "Linux dpu-bmc 5.15.50-a838e3d #1 SMP Wed Mar 27 10:44:16 UTC 2024 armv7l",
            "SystemName": "dpu-bmc"
        }
    },
    "Id": "1",
    "Links": {
        "EthernetInterfaces": [
            {
                "@odata.id": "/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0"
            }
        ]
    },
    "Name": "Manager Dedicated Network Port"
}
```

8.4.2 Enabling/Disabling LLDP on BMC

```
curl -k -u root:'<password>' -H 'Content-Type: application/json' -X PATCH https://<bmc_ip>/redfish/v1/Managers/
Bluefield_BMC/DedicatedNetworkPorts/1 -d '{"LLDPEnabled":<true/false>}'
```

The IPMI stack in OpenBMC tries to workaround this IPMI spec implementation by creating a virtual interface for the VLAN ID specified by the user and then restricting IPMI to only access the newly created virtual interface. However, this solution has side effects like the LLDP tool being unable to obtain the VLAN interface ID because the LLDP tool works only with physical interfaces.

8.5 BMC Monitoring

This section is composed of the following subpages:

8.5.1 BMC FRUs

During the initialization process, the BMC scans all supported interfaces to locate available FRU devices. Each FRU device identified by the BMC is parsed and subsequently added to the BMC's list of FRU devices.

Currently, the BMC is expected to detect the system FRU located at ID 0, as well as the BMC FRU.

The BMC FRU is available in several different revisions, each specific to the board based on its manufacturing date and type. These revisions are mainly identified by the unique product name given to each board.

FRU Field	Rev-1	Rev-2	Rev-3
FRU device description	Nvidia-BMCMezz (ID 169)	BlueField-3 DPU (ID 243)	BlueField SuperNIC (ID TBD)
Board manufacturing date	<Board-mfg-date>	<Board-mfg-date>	<Board-mfg-date>
Board manufacturer	Nvidia	Nvidia	Nvidia
Board product	Nvidia-BMCMezz	BlueField-3 DPU	BlueField SuperNIC
Board serial	<Board-serial>	<Board-serial>	<Board-serial>
Board part number	<Board-part-number>	<Board-part-number>	<Board-part-number>

8.5.1.1 BMC FRU Reading IPMI Commands

To retrieve FRU info, run:

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

Example output:

```
FRU Device Description : Nvidia-BMCMezz (ID 169)
 Board Mfg Date        : Wed Nov  8 01:41:00 2023 UTC
 Board Mfg             : Nvidia
 Board Product         : Nvidia-BMCMezz
 Board Serial          : MT2345300006
 Board Part Number     : 900-9D3B6-00CV-AAA
 Board Area Checksum   : OK
```

To print a specific FRU:

```
ipmitool -C 17 -I lanplus -H <bmc_ip> -U ADMIN -P ADMIN fru print <fru_id>
```

FRU ID of the BMC FRU EEPROM is optional and can be found using the `fru print` command.

8.5.2 Product Instance Identifier

It is possible to set a unique name for the BMC. This name may be retrieve from Redfish without a password.

8.5.2.1 Setting Product Identifier

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

8.5.2.2 Retrieving Product Identifier

- To get the system name without a password:

  ```
  curl -k -X GET https://<bmc_ip>/redfish/v1 | jq '.ServiceIdentification'
  ```

- To get the system name with a password:

  ```
  curl -k -u root:'<password>' -H 'Content-Type: application/json' -X GET https://<bmc_ip>/redfish/v1/
  Managers/Bluefield_BMC | jq '.ServiceIdentification'
  ```

8.5.3 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

8.5.3.1 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
}
```

8.5.3.2 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 formatted as `[Firmware Revision]-[ Aux Firmware Rev Info 2nd byte]` which is 23.09-1 according to this example.

8.5.4 Storage Health Monitoring

The BMC continuously monitors system resources including CPU utilization, memory usage, and storage space. When these metrics exceed configured thresholds, warning or critical event logs are automatically generated to alert administrators.

Event logs can be viewed via the Redfish API at `/redfish/v1/Systems/system/LogServices/EventLog/Entries`.

8.5.4.1 CPU Metrics

The following table defines the thresholds for CPU utilization alerts:

Metric	Alert Type	Warning	Critical	Action	Description
CPU	Upper	>80%	>95%	Log only	Total BMC CPU utilization

Metric	Alert Type	Warning	Critical	Action	Description
CPU user	Upper	>80%	>95%	Log only	CPU time in user-space applications
CPU kernel	Upper	>80%	>95%	Log only	CPU time in kernel operations

8.5.4.1.1 Metric Log Example

To check the current event log for CPU alerts:

```
curl -k -u <usr>:<password> -H 'content-type: application/json' -X GET https://<bmc_ip>/redfish/v1/Systems/
Bluefield/LogServices/EventLog/Entries
```

Example output

```
{
      "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/468",
      "@odata.type": "#LogEntry.v1_15_0.LogEntry",
      "AdditionalDataURI": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/468/attachment",
      "Created": "2026-01-19T15:57:22+00:00",
      "EntryType": "Event",
      "Id": "468",
      "Message": "CPU sensor crossed a critical high threshold going high. Reading=96.881238 Threshold=95.000000.",
      "MessageArgs": [
        "CPU",
        "96.881238",
        "95.000000"
      ],
      "MessageId": "OpenBMC.0.4.SensorThresholdCriticalHighGoingHigh",
      "Name": "System Event Log Entry",
      "Resolution": "None",
      "Resolved": false,
      "Severity": "Critical"
}
```

8.5.4.2 Memory Metrics

Metric	Alert Type	Warning	Critical	Action	Description
Memory available	Lower	<30%	<10%	Log only	Memory available for applications
Memory shared	Upper	-	>35%	Log only	Shared memory usage

8.5.4.3 Storage Metrics

The BMC system provides real-time monitoring of read-write (RW) flash usage. You can query free storage space, receive notifications when usage crosses defined thresholds, and rely on automatic cleanup when limits are exceeded.

The following table defines the thresholds for storage usage alerts. Note that percentages refer to used space:

Metric	Alert Type	Warning	Critical	Path	Action	Description
Storage RW	Lower	<10%	<5%	`/run/initramfs/rw`	Auto cleanup	Root overlay filesystem; primary writable storage for BMC runtime data and configuration changes

Metric	Alert Type	Warning	Critical	Path	Action	Description
Storage TMP	Lower	<20%	<5%	`/tmp`	Log only	Temporary files storage; used by services for transient data, cleared on reboot
Storage LOGGING	Lower	<30%	<20%	`/var/lib/logging`	Log only	Event logs and dump storage; contains Redfish logs, SEL entries, and debug dumps

8.5.4.3.1 Retrieving Free Storage Space

To check the current free RW flash space:

```
curl -k -H "X-Auth-Token: $token" -X GET https://${bmc}/redfish/v1/Managers/Bluefield_BMC/ManagerDiagnosticData'
```

Example output:

```
{
  "@odata.id": "/redfish/v1/Managers/Bluefield_BMC/ManagerDiagnosticData",
  "@odata.type": "#ManagerDiagnosticData.v1_2_0.ManagerDiagnosticData",
  "FreeStorageSpaceKiB": 1488,
  "Id": "ManagerDiagnosticData",
  "MemoryStatistics": {
    "AvailableBytes": 725983232,
    "BuffersAndCacheBytes": 170594304,
    "FreeBytes": 605347840,
    "SharedBytes": 60747776,
    "TotalBytes": 917188608
  },
  "Name": "Manager Diagnostic Data",
  "ProcessorStatistics": {
    "KernelPercent": 0.6058,
    "UserPercent": 0.5048
  },
  "ServiceRootUptimeSeconds": 1282378.351
}
```

8.5.4.3.2 Storage Cleanup Notifications

When RW flash usage exceeds 90%, a Redfish event log entry is generated to alert that manual cleanup is required.

Example log:

```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/7",
  "@odata.type": "#LogEntry.v1_15_0.LogEntry",
  "Created": "2025-09-15T13:30:43+00:00",
  "EntryType": "Event",
  "Id": "7",
  "Message": "Processes consuming HIGH Resource Storage_RW are 91%",
  "MessageArgs": [
    "Storage_RW",
    "91%"
  ],
  "MessageId": "OpenBMC.0.4.BMCSystemResourceInfo",
  "Name": "System Event Log Entry",
  "Resolution": "None.",
  "Resolved": false,
  "Severity": "OK"
}
```

8.5.4.3.3 Automatic Cleanup

When RW flash usage exceeds 95%, the BMC automatically purges space by deleting:

- All dump files
- All event logs
- Files in home directories
- Files in system log directories

Exceeding 99% RW flash usage can make BMC functionality unstable and impede automatic cleanup.

After automatic cleanup, a Redfish event log entry is generated. For example:


```
{
  "@odata.id": "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/3",
  "@odata.type": "#LogEntry.v1_15_0.LogEntry",
  "Created": "2025-09-24T10:40:34+00:00",
  "EntryType": "Event",
  "Id": "3",
  "Message": "RWFS cleanup completed.",
  "Modified": "2025-09-22T10:40:34+00:00",
  "Name": "System Event Log Entry",
  "Resolved": false,
      "Severity": "OK"
    }
```


8.6 BMC Reset Control

This section is comprised of the following subpages:

- Factory Reset BMC
- Reset or Reboot BMC

8.6.1 Factory Reset BMC

The following commands factory reset the BMC configuration.

8.6.1.1 1. 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.ResetToDefaults -d '{"ResetToDefaultsType": "ResetAll"}'
```

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".

After running factory reset, the BIOS configuration attributes list is updated only after rebooting the BlueField as the list gets its values from UEFI as BlueField is booting and Redfish is enabled.

8.6.1.2 2. 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.

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

```
ipmitool mc reset cold
```

In factory reset the time is also restarting, and time sync to the real time start after some of the services start. Because of that some of the services will show that there active time to be start at 1970.

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".

8.6.1.3 3. Secure Factory Reset

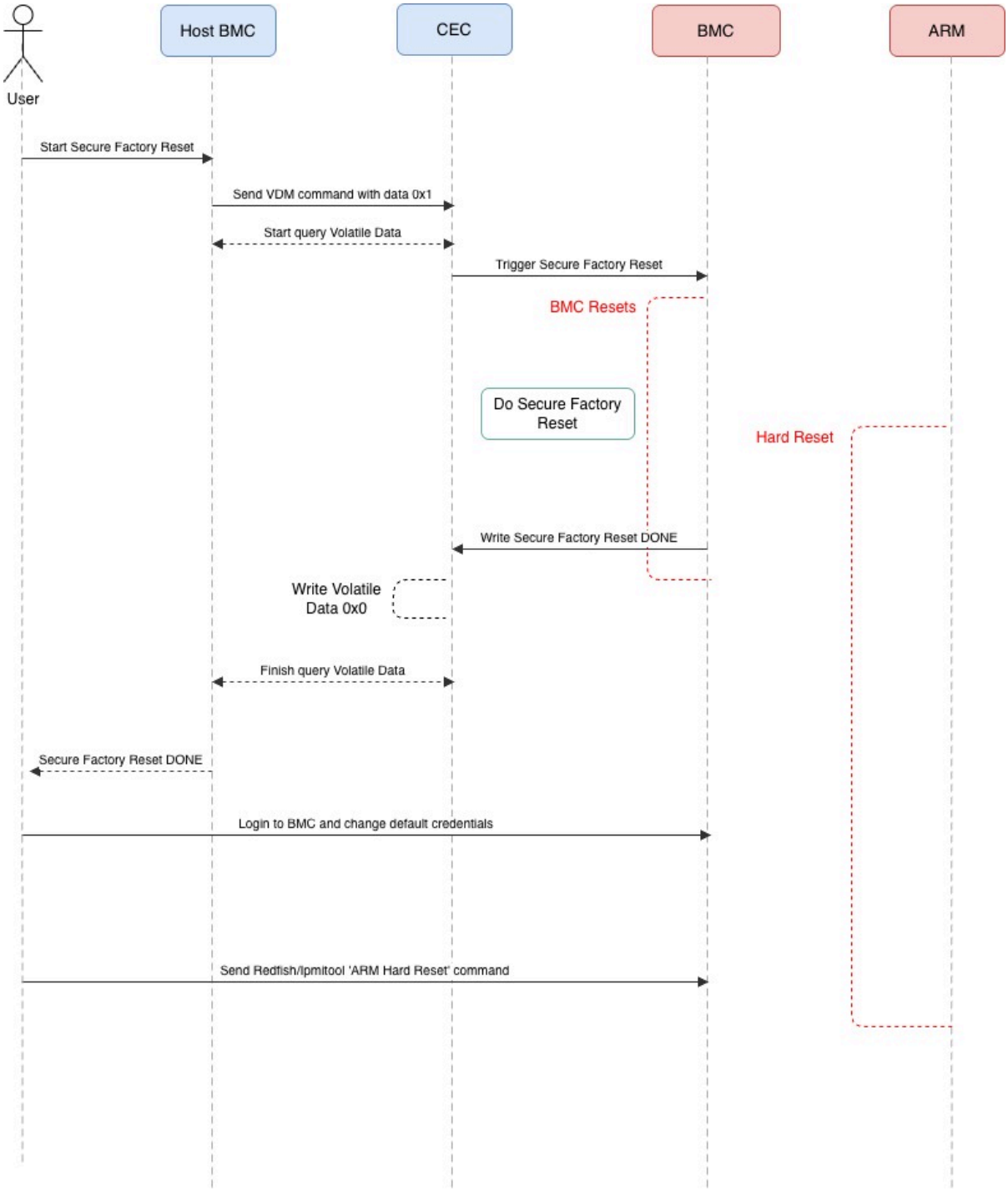
This feature is considered BETA for 26.04 release.

When the BMC was compromised or is not reachable for any reason, there is an option to ensure a factory reset takes place.

This Feature is relevant for BlueField3 only.

The Secure Factory Reset will:

1. Do regular Factory Reset of the BMC.
2. Clear the Config Partition.
3. Hold the DPU's Arm in reset, until an Arm Hard Reset command is received in the BMC.



Prerequisites

1. Host-BMC which is connected to the CEC over SMBus through GoldenFinger.
2. Implementation of the following VDM command over MCTP stack, on the Host-BMC side:
   < Placeholder for Glacier's VDM doc with the command >

1. Initiate the flow by sending the VDM command of 'Secure Factory Reset Request' from the Host-BMC to the CEC.
2. Wait for the BMC to boot, check the status of the Secure Factory Reset from the Host-BMC by polling on the Volatile Data returned from "Secure Factory Reset Query Volatile Data" command.
3. Once the Volatile Data is 0x00, that is the indication that Secure Factory Reset is complete.
4. From Redfish or IPMITool, run the ARM Hard Reset command to bring back the DPU's Arm.

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".

8.6.2 Reset or Reboot BMC

8.6.2.1 Rebooting 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
```

The `ResetType` can be following actions:

1. a. GracefulRestart - stop running services before reboot BMC;
   b. ForceRestart - reboot BMC without stop running services(serval times force restart will lead to ERoT stuck, need do power cycle to recover);
   c. GracefulShutdown - shutdown BMC (A following system power event should be issued, otherwise the BMC will boot up).

8.6.2.2 Rebooting BMC IPMI Command

```
ipmitool mc re cold
```

8.7 BMC Networking

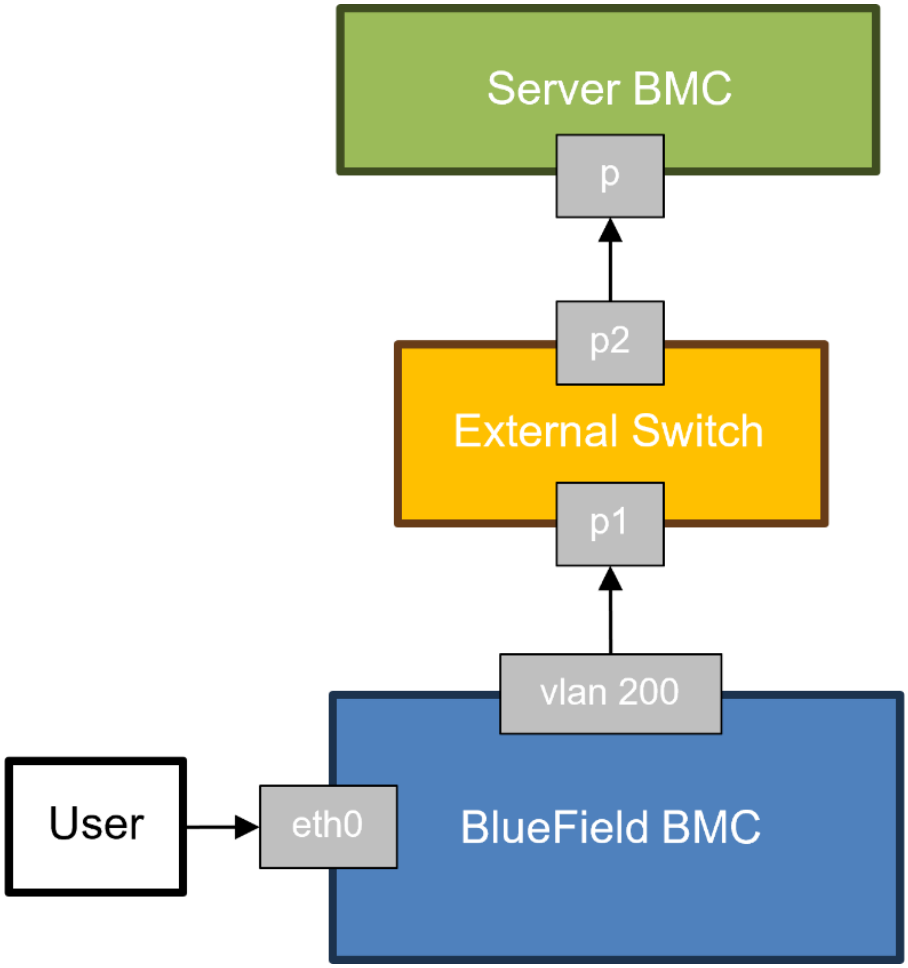
This section is comprised of the following subpages:

- Guest Tunnel
- Network Protocol Support

8.7.1 Guest Tunnel

The NVIDIA® BlueField® BMC is capable of establishing a designated VLAN interface to forward IPMI or HTTPS traffic from an external source to a specific IP address within that VLAN. Users can set a

specific IP address for their server BMC, which allows for the management of their server BMC via the BlueField BMC.



To enable this feature, users must configure their network according to the following:

- Assign the remote server's BMC IP address as 192.168.1.10 to enable traffic forwarding
- The VLAN ID for the guest tunnel is 200, thus the external switch linked to the BlueField RJ45 port (OOB) must be set up to accept packets tagged with VLAN 200

The BlueField BMC currently supports the following ports, which act as source ports to accept messages sent by users and forward them to the server BMC within the guest tunnel:

- 8443 – Port on BlueField BMC for accepting HTTPS messages
- 8623 – Port on BlueField BMC for accepting IPMI messages

> The guest tunnel is intended solely for debugging purposes. Users should refrain from sending large amounts of network traffic through the guest tunnel, as it may impact the performance of the BlueField BMC.

8.7.1.1 Querying Guest Tunnel Status

netfunc	cmd	data	Notes
0x3E	0xFD	0x0	Displays the current configuration: • 0x01 - Disabled • 0x02 - Enabled

8.7.1.2 Disabling Guest Tunnel

netfunc	cmd	data	Notes
0x3E	0xFD	0x1	On success, returns: • 0x01 - Guest Tunnel is set to Disabled

8.7.1.3 Enabling Guest Tunnel

netfunc	cmd	data	Notes
0x3E	0xFD	0x2	On success, returns: • 0x02 - Guest Tunnel is set to Enabled

Example for enabling the guest tunnel on BlueField BMC:

```
#bmc> ipmitool raw 0x3e 0xfd 0x2
```

Example for sending the Redfish command to the guest tunnel:

```
#localhost> curl -k -u <bluefield_bmc_username>:<bluefield_bmc_password> -H 'content-type: application/json' -X GET
https://<bluefield_bmc_ip>:8443/redfish/v1/Systems/Bluefield
```

Example for sending the IPMI command to the guest tunnel:

```
#localhost> ipmitool -p 8623 -C 17 -I lanplus -H <bluefield_bmc_ip> -U <bluefield_bmc_username> -P
<bluefield_bmc_password> mc info
```

After receiving the responses from the Redfish and IPMI commands, the content of these responses originates from the server BMC, not the BlueField BMC.

8.7.2 Network Protocol Support

> ⚠ To obtain the BMC MAC address, refer to the board label affixed to the NVIDIA® BlueField® device.

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

8.7.2.1 Network Management Redfish Commands

8.7.2.1.1 Getting Network Protocol Configuration

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

8.7.2.1.2 Getting Interface Configuration

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

8.7.2.1.3 Enabling/Disabling 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`.

> **Disabling the `eth0` interface on the BlueField BMC prevents OOB network functionality on the BMC. This inhibits the ability to execute any Redfish or IPMI commands through the network.**

8.7.2.1.4 Configuring Static IPv4 Address

```
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"}]}'
```

8.7.2.1.5 Deleting Static IPv4 Address

```
curl -k -u root:'<password>' -X PATCH https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0 -d
'{"IPv4StaticAddresses": [null]}'
```

8.7.2.1.6 Enabling/Disabling IPv4 DHCP

```
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`.

8.7.2.1.7 Configuring Static DNS Server IPv4 and IPv6

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

8.7.2.1.8 Configuring Static IPv6 Address

```
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}]}'
```

8.7.2.1.9 Enabling/Disabling IPv6 DHCP

```
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:

- `Enabled` - DHCPv6 is enabled for this interface
- `Disabled` - DHCPv6 is disabled for this interface

```
curl -k -u root:'<password>' -X PATCH https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0 -d
'{"StatelessAddressAutoConfig": {"IPv6AutoConfigEnabled": "<state>"}}'
```

Where `<state>` can be:

- `true` - Indicate IPv6 stateless address autoconfiguration (SLAAC) is enabled for this interface
- `false` - Indicate IPv6 stateless address autoconfiguration is disabled for this interface

8.7.2.1.10 Enabling/Disabling NTP

```
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`.

8.7.2.1.11 Configuring Static NTP Server IP

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

8.7.2.2 Network Management IPMI Commands

The following subsections list the available network IPMI commands.

8.7.2.2.1 Configuring IPv4 Mode

The following command sets LAN channel 1 IP config mode to static or DHCP which corresponds to network interface `eth0`.

```
ipmitool lan set 1 ipsrc <mode>
```

Where `<mode>` can be `static` or `dhcp`.

8.7.2.2.2 Configuring IPv6 Mode

The following command sets LAN channel 1 IP config mode to static or DHCP which corresponds to network interface `eth0`.

```
ipmitool lan6 set 1 rtr_cfg <mode>
```

Where `<mode>` can be `static` or `dynamic`. `both` is unsupported.

8.7.2.2.3 Adding IPv4 Address

The following commands add IPv4 address, default gateway, and netmask to the network interface `eth0`.

- IP address:

  ```
  ipmitool lan set 1 ipaddr <ip-address>
  ```

- Default gateway:

  ```
  ipmitool lan set 1 defgw ipaddr <ip-address>
  ```

- Netmask:

  ```
  ipmitool lan set 1 netmask <netmask>
  ```

  ⚠ IPMI supports only a single static IP address. If multiple static IP addresses are configured on the system, the new netmask will be applied to only one of them.

8.7.2.2.4 Getting IPv4 Config

The following command gets IPv4 network config for channel 1 which corresponds to the network interface `eth0`.

```
ipmitool lan print 1
```

8.7.2.2.5 Setting IPv6 Address

The following command adds IPv6 address to the network interface `eth0`.

```
ipmitool lan6 set 1 nolock static_addr 0 enable <ipv6-address> 64
```

8.7.2.2.6 Getting IPv6 Config

The following command gets IPv6 network config for channel 1 which corresponds to the network interface `eth0`.

```
ipmitool lan6 print 1
```

8.7.2.2.7 Getting DNS Server

```
ipmitool raw 0x32 0x6B
```

Output:

```
0b 31 30 2e 31 35 2e 31 32 2e 36 37
```

This output corresponds to `10.15.12.67`.

8.7.2.2.8 Adding 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
```

This output corresponds to `10.15.12.67`.

8.7.2.2.9 Getting 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
- `31 2e 69 6e 2e 70 6f 6f 6c 2e 6e 74 70 2e 6f 72 67` - NTP server address byte stream which corresponds to `1.in.pool.ntp.org`

8.7.2.2.10 Adding 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 which corresponds to `1.in.pool.ntp.org`

8.7.2.2.11 Enabling NTP Time Sync

The following command enables time sync to NTP server.

```
ipmitool raw 0x32 0xA8 0x02 0x01
```

Where:

- `0x01` - enable NTP

8.7.2.2.12 Disabling NTP Time Sync

The following command disables time sync to NTP server.

```
ipmitool raw 0x32 0xA8 0x02 0x00
```

Where:

- `0x00` - disable NTP

8.7.2.2.13 Configuring Router IPv6 Mode

The following command sets router mode to static or DHCP.

```
ipmitool lan6 set 1 rtr_cfg <mode>
```

Where `<mode>` can be:

- `static`
- `Dynamic`

⚠ Configuring static mode also requires setting the static router IP and static router MAC address.

⚠ Router prefix can only be 0.

8.7.2.2.14 Configuring IPv6 Static Router IP

The following command sets the IPv6 address for the static router.

```
ipmitool raw 0x0c 0x01 0x01 0x41 <ip-hex>
```

Where:

- `<ip-hex>` - the IP address

8.7.2.2.15 Configuring IPv6 Static Router MAC

The following command sets the MAC address for the static router.

```
ipmitool raw 0x0c 0x01 0x01 0x42 <mac-hex>
```

Where:

- `<mac-hex>` - the IP MAC address

8.8 BMC Redfish

This section is comprised of the following subpages:

8.8.1 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.

8.8.1.1 Adding Numeric Trigger

Adds a numeric trigger to the BMC.

```
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":["<>"]}'
```

8.8.1.2 Deleting Numeric Trigger

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X DELETE https://<bmc_ip>/redfish/v1/
TelemetryService/Triggers/<trigger-name>
```

8.8.2 Redfish Certificate Management

Certificate management actions—such as retrieving certificate information or performing atomic certificate replacement—are accessible through the `CertificateService` resource.

The `CertificateLocations` resource provides an inventory of all certificates managed by the service.

For additional details, refer to the *Redfish Certificate Management White Paper*.

8.8.2.1 Common Certificate Management Commands

8.8.2.1.1 Getting Certificate Locations

Inventory of all certificates the service is managing.

```
curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/CertificateService/CertificateLocations
```

8.8.2.2 Root CA Management Commands

8.8.2.2.1 List Root CA

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

8.8.2.2.2 Getting Certificate Information

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

8.8.2.2.3 Installing Root CA Certificate

```
curl -k -u root:'<password>' -X POST https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Truststore/Certificates -d
@rootca.json
```

8.8.2.2.4 Replacing Existing Root CA Certificate

```
curl -k -u root:'<password>' -X PATCH https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Truststore/Certificates/1
-d @rootca.json
```

8.8.2.2.5 Root CA Certificate Creation and Replacement

1. Generate Root CA certificate:

```
cat > root-ca.cnf << EOF
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req
prompt = no

[req_distinguished_name]
C = <country>
ST = <state>
L = <location>
O = OpenBMC
OU = bmcweb
CN = <common_name>

[v3_req]
basicConstraints = critical,CA:true
keyUsage = critical,keyCertSign,cRLSign
subjectKeyIdentifier = hash
EOF

# Generate root CA key
openssl genrsa -out root-ca-key.pem <key_size>

# Generate root CA certificate
openssl req -x509 -new -nodes \
    -key root-ca-key.pem \
    -sha256 -days <validity_days> \
    -out root-ca-cert.pem \
    -config root-ca.cnf \
    -extensions v3_req
```

2. Create a JSON file for the root CA certificate add

```
{
    "CertificateString": "<cert_string>",
    "CertificateType": "PEM"
}
```

3. Install the root CA certificate (can have more then 1).

```
curl -k -u root:'<password>' -X POST https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Truststore/
Certificates -d @rootca.json
```

8.8.2.3 Server Certificate Management Commands

8.8.2.3.1 Getting Certificate Information

```
curl -k -u root:'<password>' -X GET https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/NetworkProtocol/HTTPS/
Certificates/1
```

8.8.2.3.2 Replacing Existing Certificate

```
curl -k -u root:'<password>' -X POST https://<bmc_ip>/redfish/v1/CertificateService/Actions/
CertificateService.ReplaceCertificate -d @certificate.json
```

8.8.2.3.3 Generating CSR

Generate certificate signing request (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
```

8.8.2.3.4 Installing 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
```

8.8.2.3.5 Example for CSR Generation, Certificate Creation and Replacement

1. Configure your CA to include at least the following extensions for the signed TLS server certificates:

   ```
   basicConstraints = CA:FALSE
   keyUsage = nonRepudiation, digitalSignature, keyEncipherment
   subjectAltName = IP:192.168.240.1
   ```

   The extension `subjectAltName = IP:192.168.240.1` is mandatory.

2. Create a JSON containing the subject data for the BlueField BMC to use when creating the CSR. For example:

   ```
   {
       "City": "<city>",
       "CertificateCollection": {
           "@odata.id": "/redfish/v1/Managers/Bluefield_BMC/NetworkProtocol/HTTPS/Certificates/"
       },
       "CommonName": "bmc0123456789.mycompany.com",
       "Country": "<country>",
       "Organization": "<company_name>",
       "OrganizationalUnit": "<my_org>",
       "State": "<state>",
       "KeyPairAlgorithm": "EC"
   }
   ```

3. Generate a certificate signing request using the Redfish Certificate Management#forth command in the table above and the JSON file created in the previous step:

   The BMC replies with a JSON containing the 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
   {
     "CSRString": "-----BEGIN CERTIFICATE REQUEST-----\<CSR_DATA>\n-----END CERTIFICATE REQUEST-----\n",
     "CertificateCollection": {
       "@odata.id": "/redfish/v1/Managers/Bluefield_BMC/NetworkProtocol/HTTPS/Certificates/"
     }
   }
   ```

4. Extract the CSR string from the JSON and sign the CSR using your CA. For example, this is how to include the required extensions to the signed TLS server certificates:

```
openssl x509 -req -in bmc.csr -CA CA-cert.pem -CAkey CA-key.pem -CAcreateserial -out bmc.crt -days 3650
-sha384 -extfile exfile.txt
```

Where:

- `bmc.csr` contains the CSR string from the previous step
- `CA-cert.pem` contains the CA certificate to be used to sign the CSR
- `CA-key.pem` contains the CA private key
- `extfile.txt` contains the extensions mentioned in the first step (`basicConstraints`, `keyUsage`, and `subjectAltName`)
- `bmc.crt` is the output file which will contain the BMC certificate signed by the CA

5. Create a JSON file for the BlueField BMC signed TLS server certificate data:

```
{
    "CertificateString": "-----BEGIN CERTIFICATE-----\n<bmc.crt-data>\n-----END CERTIFICATE-----",
    "CertificateType": "PEM",
    "CertificateUri":
    {
        "@odata.id": "/redfish/v1/Managers/Bluefield_BMC/NetworkProtocol/HTTPS/Certificates/1"
    }
}
```

6. Replace the BMC certificate using the Redfish Certificate Management#third command in the table above and the JSON created in the previous step.

```
curl -k -u root:'<password>' -X POST https://<bmc_ip>/redfish/v1/CertificateService/Actions/
CertificateService.ReplaceCertificate -d @certificate.j
```

9 NIC Subsystem Management

⚠ This content is relevant for NVIDIA® BlueField®-3 devices only.

9.1 Redfish NIC Subsystem Management

9.1.1 Configuring BlueField Mode of Operation

Refer to "BlueField Modes of Operation Configuration" for information.

9.1.2 Getting Host RShim

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

9.1.3 Enabling Host RShim

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

9.1.4 Disabling Host RShim

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

9.1.5 Getting Strap Options

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

9.1.6 Host Privileges Configuration

This resource manages the security privileges assigned to the host interface. It allows administrators to restrict the host's ability to modify device configurations or access sensitive parameters.

```
https://<bmc_ip>/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/Oem/Nvidia/HostPrivilegeConfig
```

9.1.6.1 Privilege Modes (Presets)

The `PrivilegeMode` attribute acts as a master switch, applying a predefined set of permissions.

Mode	Description
"Privileged" (default)	Grants full access. The host can modify firmware, flash, and global parameters.

Mode	Description
"Restricted"	Locks down the host. Prevents modification of firmware, flash, and global parameters. RSHIM and Tracer access are disabled.

9.1.6.2 Configuration Breakdown by Mode

The following table shows exactly which permissions are enabled or disabled for each mode:

Setting	Privileged Mode	Restricted Mode
`HostPrivilegeLevel`	`"Privileged"`	`"Restricted"`
`FirmwareUpdate`	`"Enabled"`	`"Disabled"`
`FlashAccess`	`"Enabled"`	`"Disabled"`
`GlobalParametersAccess`	`"Enabled"`	`"Disabled"`
`HostParametersAccess`	`"Enabled"`	`"Disabled"`
`InternalCPUAccess`	`"Enabled"`	`"Disabled"`
`NicReset`	`"Enabled"`	`"Disabled"`
`PccUpdate`	`"Enabled"`	`"Disabled"`
`PortAccess`	`"Enabled"`	`"Disabled"`
`ManagementInterfaceEnabled`	`true`	`false`
`PortOwnerEnabled`	`true`	`false`
`ReadCountersEnabled`	`true`	`false`
`TracerEnabled`	`true`	`false`

9.1.6.3 Privilege Settings Definitions

This table defines the specific behavior controlled by each permission setting.

Setting	Description	Options	Default
`HostPrivilegeLevel`	Defines the overarching privilege tier for the host. When set to `Restricted`, it locks down firmware and global parameter modifications. ⚠ To transition to `Restricted` mode, any granular properties (like `FirmwareUpdate` or `FlashAccess`) set to `Enabled` must be changed to default beforehand or simultaneously.	`"Privileged"`, `"Restricted"`	`"Privileged"`

Setting	Description	Options	Default
`FlashAccess`	Permission to perform any device flash access.	`"Default"`, `"Enable"`, `"Disable"`	`"Default"`
`PccUpdate`	Permission to update the Programmable Congestion Control (PCC) algorithm.	`"Default"`, `"Enable"`, `"Disable"`	`"Default"`
`FirmwareUpdate`	Permission to perform firmware updates.	`"Default"`, `"Enable"`, `"Disable"`	`"Default"`
`NicReset`	Permission to perform a NIC reset.	`"Default"`, `"Enable"`, `"Disable"`	`"Default"`
`GlobalParametersAccess`	Permission to access global non-volatile (NV) parameters.	`"Default"`, `"Enable"`, `"Disable"`	`"Default"`
`HostParametersAccess`	Permission to access host NV parameters.	`"Default"`, `"Enable"`, `"Disable"`	`"Default"`
`PortAccess`	Permission to access port NV parameters.	`"Default"`, `"Enable"`, `"Disable"`	`"Default"`
`InternalCPUAccess`	Permission to access internal CPU NV parameters.	`"Default"`, `"Enable"`, `"Disable"`	`"Default"`
`ManagementInterfaceEnabled`	Controls RShim function. If `false`, the host cannot access embedded CPU registers.	`true`, `false`	—
`PortOwnerEnabled`	Controls port ownership. If `false`, the host cannot become the port owner.	`true`, `false`	—
`ReadCountersEnabled`	Controls physical counter access. If `false`, the host cannot read physical port counters.	`true`, `false`	—
`TracerEnabled`	Controls tracer ownership. If `false`, the host cannot own the Tracer.	`true`, `false`	—

9.1.6.4 Example Usage

The following example demonstrates a `GET` request to retrieve the current privilege settings.

- Request:

```
curl -u 'root':'<password>' -X GET https://<bmc_ip>/redfish/v1/Chassis/Card1/NetworkAdapters/
NvidiaNetworkAdapter/Oem/Nvidia/HostPrivilegeConfig
```

- Response:


```
{
  "@Redfish.Settings": {
    "@odata.type": "#Settings.v1_3_5.Settings",
    "SettingsObject": {
      "@odata.id": "/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/Oem/Nvidia/
HostPrivilegeConfig/Settings"
    }
  },
  "@odata.id": "/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/Oem/Nvidia/
HostPrivilegeConfig",
  "@odata.type": "#NvidiaHostPrivilegeConfig.v1_0_0.NvidiaHostPrivilegeConfig",
  "Id": "HostPrivilegeConfig",
  "Name": "Host Privilege Configuration",
  "PrivilegeMode": "Privileged",
  "PrivilegeSettings": {
    "FirmwareUpdate": "Enabled",
    "FlashAccess": "Enabled",
    "GlobalParametersAccess": "Enabled",
    "HostParametersAccess": "Enabled",
    "HostPrivilegeLevel": "Privileged",
    "InternalCPUAccess": "Enabled",
    "ManagementInterfaceEnabled": true,
    "NicReset": "Enabled",
    "PccUpdate": "Enabled",
    "PortAccess": "Enabled",
    "PortOwnerEnabled": true,
    "ReadCountersEnabled": true,
    "TracerEnabled": true
  }
}
```


9.2 Setting Host Privilege Configuration

To modify host privileges, send a `PATCH` request to the `Settings` URI.

```
PATCH https://<bmc_ip>/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/Oem/Nvidia/
HostPrivilegeConfig/Settings
```

The `/Settings` path displays pending values. These changes do not take effect immediately.

9.2.1 Configuration Examples

9.2.1.1 View Pending Settings

Before making changes, you can verify the current pending configuration.

- Request

```
curl -u 'root':'<password>' -X GET https://<bmc_ip>/redfish/v1/Chassis/Card1/NetworkAdapters/
NvidiaNetworkAdapter/Oem/Nvidia/HostPrivilegeConfig/Settings
```

- Response


```
{
  "@odata.id": "/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/Oem/Nvidia/
HostPrivilegeConfig/Settings",
  "@odata.type": "#NvidiaHostPrivilegeConfig.v1_0_0.NvidiaHostPrivilegeConfig",
  "Id": "Settings",
  "Name": "Host Privilege Configuration Settings",
```

```
  "PrivilegeMode": "Privileged",
  "PrivilegeSettings": {
    "FirmwareUpdate": "Enabled",
    "FlashAccess": "Enabled",
    "GlobalParametersAccess": "Enabled",
    "HostParametersAccess": "Enabled",
    "HostPrivilegeLevel": "Privileged",
    "InternalCPUAccess": "Enabled",
    "ManagementInterfaceEnabled": true,
    "NicReset": "Enabled",
    "PccUpdate": "Enabled",
    "PortAccess": "Enabled",
    "PortOwnerEnabled": true,
    "ReadCountersEnabled": true,
    "TracerEnabled": true
  }
}
```

9.2.1.2 Set Privilege Mode

To apply a high-level preset (Privileged or Restricted):

```
curl -u 'root':'<password>' -X PATCH -H "Content-Type: application/json" \
https://<bmc_ip>/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/Oem/Nvidia/HostPrivilegeConfig/
Settings \
-d '{"PrivilegeMode":"Privileged"}'
```

9.2.1.3 Set Specific Properties

To apply granular permissions. Note that these are nested within the `PrivilegeSettings` object:

```
curl -u 'root':'<password>' -X PATCH -H "Content-Type: application/json" \
https://<bmc_ip>/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/Oem/Nvidia/HostPrivilegeConfig/
Settings \
-d '{
    "PrivilegeSettings": {
        "NicReset": "Default",
        "PccUpdate": "Default",
        "PortAccess": "Default",
        "FirmwareUpdate": "Default",
        "FlashAccess": "Disabled",
        "GlobalParametersAccess": "Disabled",
        "HostParametersAccess": "Disabled",
        "InternalCPUAccess": "Disabled",
        "HostPrivilegeLevel": "Privileged"
    }
}'
```

9.2.2 Logic and Constraints

> ⚠ Every change requires a System Power Cycle to take effect.

9.2.3 Parameter Precedence

If a conflict arises between granular access controls and global/functional flags, the granular access controls take precedence.

Category	Parameters	Precedence
Granular Access Controls	`HostParametersAccess`, `PortAccess`, `GlobalParametersAccess`, `InternalCPUAccess`, `PccUpdate`, `FirmwareUpdate`, `FlashAccess`, `NicReset`	High (wins conflicts)

Category	Parameters	Precedence
Global and Functional Flags	`HostPrivilegeLevel`, `ManagementInterfaceEnabled`, `PortOwnerEnabled`, `ReadCountersEnabled`, `TracerEnabled`	Low

9.2.4 Transitioning to Restricted Mode

If any of the following properties are currently set to `Enabled`, you cannot change `HostPrivilegeLevel` to `Restricted` immediately:

- `GlobalParametersAccess`
- `PortAccess`
- `InternalCPUAccess`
- `NicReset`
- `FirmwareUpdate`
- `FlashAccess`

You must first (or simultaneously) set the conflicting property to `Default` before the system accepts the `Restricted` level.

9.3 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 BlueField 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:

netfunc	cmd	data	Description
`0x32`	`0x9C`	N/A	Get SmartNIC mode. Prints current configuration: `INTERNAL_CPU_OFFLOAD_ENGINE`. • `00` - Disabled • `01` - Enabled

netfunc	cmd	data	Description
`0x32`	`0x9D`	Byte0	Set SmartNIC mode (`INTERNAL_CPU_OFFLOAD_ENGINE`) to `Byte0` . Supported values: • `00` - Disabled • `01` - Enabled
`0x32`	`0x9E`	N/A	Get host access. Prints current `HOST_PRIV_RSHIM` . • `00` - Disabled • `01` - Enabled
`0x32`	`0x9F`	Byte0	Set host access. Sets `HOST_PRIV_RSHIM` to `Byte0` . Supported values: • `00` - Disabled • `01` - Enabled

<table>
<tr><th>netfunc</th><th>cmd</th><th>data</th><th>Description</th></tr>
<tr><td><code>0x32</code></td><td><code>0xA2</code></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><code>VERSION</code></td></tr>
<tr><td>1</td><td><code>DISABLE_INBAND_RECOVER_VALUE</code></td></tr>
<tr><td>2</td><td><code>PRIMARY_IS_PCORE_1_VALUE</code></td></tr>
<tr><td>3</td><td><code>2PCORE_ACTIVE_VALUE</code></td></tr>
<tr><td>4</td><td><code>SOCKET_DIRECT_VALUE</code></td></tr>
<tr><td>5</td><td><code>PCI_REVERSAL_VALUE</code></td></tr>
<tr><td>6</td><td><code>PCI_PARTITION_1_VALUE</code></td></tr>
<tr><td>7</td><td><code>PCI_PARTITION_0_VALUE</code></td></tr>
<tr><td>8</td><td><code>OSC_FREQ_1_VALUE</code></td></tr>
<tr><td>9</td><td><code>OSC_FREQ_0_VALUE</code></td></tr>
<tr><td>10</td><td><code>CORE_BYPASS_N_VALUE</code></td></tr>
<tr><td>11</td><td><code>FNP_VALUE</code></td></tr>
<tr><td>12</td><td><code>DISABLE_INBAND_RECOVER_VALUE</code></td></tr>
<tr><td>13</td><td><code>PRIMARY_IS_PCORE_1_MASK</code></td></tr>
<tr><td>14</td><td><code>2PCORE_ACTIVE_MASK</code></td></tr>
<tr><td>15</td><td><code>SOCKET_DIRECT_MASK</code></td></tr>
<tr><td>16</td><td><code>PCI_REVERSAL_MASK</code></td></tr>
<tr><td>17</td><td><code>PCI_PARTITION_1_MASK</code></td></tr>
<tr><td>18</td><td><code>PCI_PARTITION_0_MASK</code></td></tr>
<tr><td>19</td><td><code>OSC_FREQ_1_MASK</code></td></tr>
<tr><td>20</td><td><code>OSC_FREQ_0_MASK</code></td></tr>
<tr><td>21</td><td><code>CORE_BYPASS_N_MASK</code></td></tr>
<tr><td>22</td><td><code>FNP_MASK</code></td></tr>
</table>
Each state is represented by binary byte in order.
Supported values:
• <code>00</code> - Disabled
• <code>01</code> - Enabled</td></tr>
</table>

netfunc	cmd	data	Description
0x32	0xA3	N/A	Get SmartNIC OS State. • 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

9.3.1 Setting Operation Mode

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

9.3.2 Enabling/Disabling RShim from Host

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

10 Table of Common Redfish Commands

Capability	Redfish	IPMItool	Supported in NIC Mode? [Y/N]
Changing the default BMC password	Changing default password using Redfish	N/A	Yes
Changing the default UEFI password	Changing UEFI Password	N/A	Yes
Enabling/disabling secure boot	Setting Secure Boot State	N/A	N/A
Updating BMC firmware	BMC and CEC firmware update	N/A	Yes
Updating BlueField BFB	Pushing BFB from BMC to BlueField Arm	N/A	Yes
Configuring BlueField to network boot from the out-of-band interface first	Boot Config Using Redfish	Boot Config Using IPMI	N/A
Resetting BlueField	Reset control	Reset control	Partial
Resetting BlueField BMC	Reset control using Redfish	Reset control using IPMI	Yes
Factory reset	Factory Reset Redfish Command	Factory Reset IPMI Command	Yes
Getting BlueField versions	System inventory	N/A	No
Getting BlueField BMC versions	Retrieving BMC version using Redfish	Retrieving BMC version using IPMI command	Yes
Getting high-speed ports MAC addresses for mapping BlueField's Ethernet devices	Chassis Card1 NetworkAdapters	List of IPMI Supported FRUs	No
BlueField monitoring (SEL, FRU, etc.)	Monitoring	Monitoring	No
User management	User management Redfish commands	User management IPMI commands	Yes
Enabling secure boot with customer keys	BIOS secure boot configuration	N/A	N/A
Enabling/disabling zero-trust mode	Disable host RShim	Enable/disable RShim from Host	Yes - BlueField-3 only
Enabling RShim from BlueField BMC	Enable RShim on BlueField BMC	Enable RShim	Yes
Changing BlueField mode	Redfish NIC Subsystem Management	Setting operation mode	Yes - BlueField-3 only
Partial BFB update (ATF/UEFI)	Deploying Software Using BFB	NA	Yes

11 Unsupported BMC Functionalities in NIC Mode

Affected Capability	Effect	Section Reference
Sensors and SDRs	DPU sensors: • `bluefield_tempj` • `ddr_temp` • `p0_temp` • `p0_link` • `p1_temp` (if present) • `p1_link` (if present) • `rtc_voltage` • `power_envelope` • `soc_power` Available sensors/FRU may be are retrievable via `ipmitool -I ipmb sdr list all`. This impacts Redfish `Sensors` schema and IPMItool outputs like `ipmitool sensors` and `ipmitool sdr`.	• BMC Sensor Data
Network interfaces	DPU network ports: • `eth0` • `eth1` (if present) • `oob0` Impact: • Redfish schemas - • `redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/` • `redfish/v1/Systems/Bluefield/EthernetInterfaces/` •	• BlueField Host Network Interface
Firmware inventory	Objects with prefix `DPU_*` do not show the `Version` in the Redfish `FirmwareInventory` schema	• System Inventory
Card1 schema	Redfish Chassis Card1 schema: Information like manufacturer, model, versions, and assemblies from BlueField FRU is affected.	• DPU Chassis
FRU	DPU FRU files: Extracted using IPMB. Affects FRU 0 on the BMC, OEM FRU, and functionalities using BlueField FRU. OEM FRU entries: `dmidecode` reads fail, leading to default values.	System FRU
`bmc-set-time.service`	The service syncs BMC time with DPUs when BMC NTP is unavailable	N/A

Affected Capability	Effect	Section Reference
Arm OS graceful reset	Unavailable: `ipmitool raw 0x32 0xA1 0x02` and Redfish `redfish/v1/Systems/Bluefield/Actions/ComputerSystem.Reset -d {"ResetType": "GracefulShutdown"}`.	Reset Control
Redfish system logs	DPU OS logs are missing: • `/var/log/dmesg` • `/var/log/lastlog` • `/var/log/wtmp`	BMC and BlueField Logs
Base GUID, base MAC, and description from Redfish OEM	Fields in `Systems/Bluefield/Oem/Nvidia` Redfish schema are affected as they depend on IPMB	DPU Information

12 Appendixes

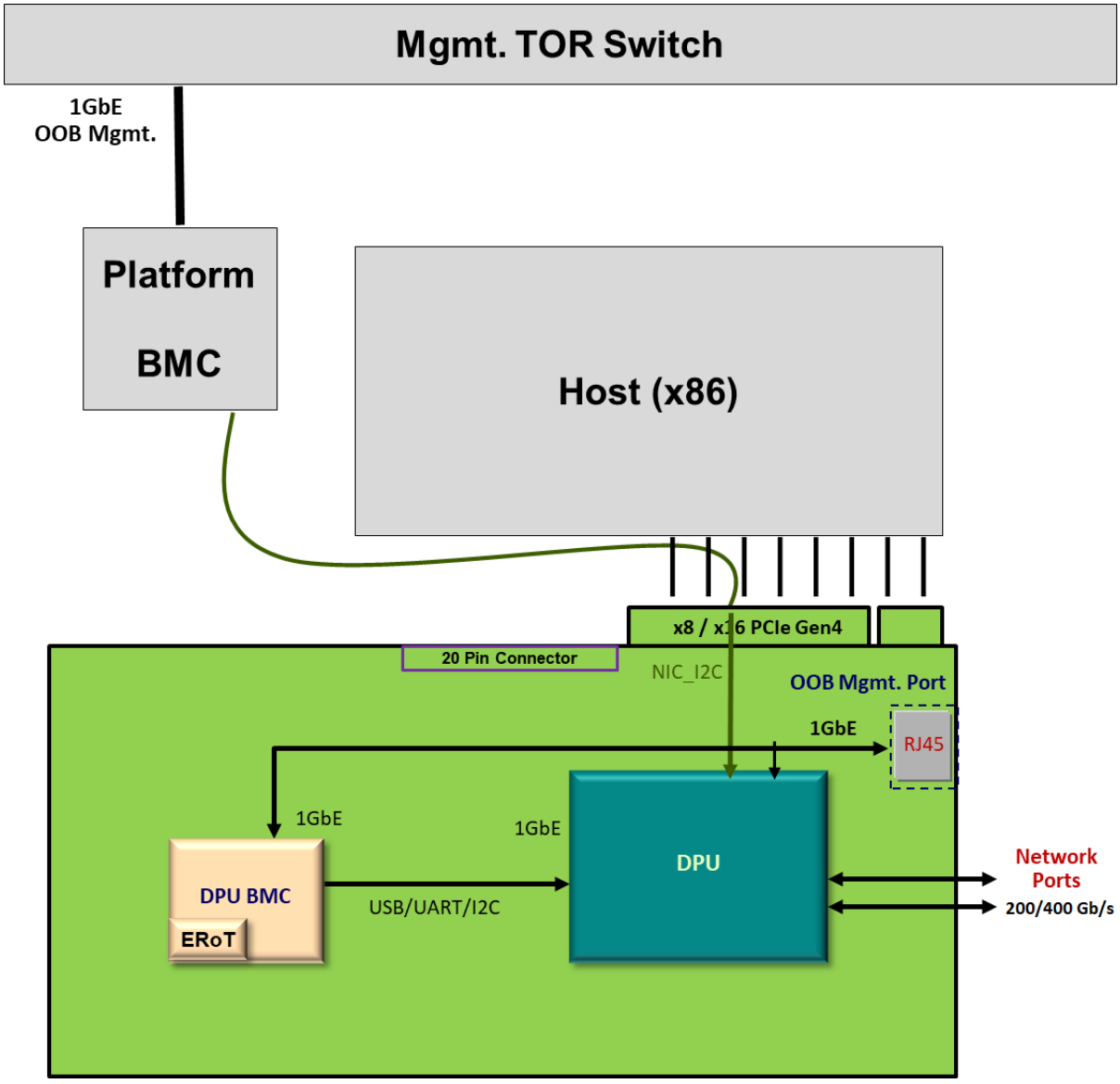
The following appendixes are available:

12.1 Appendix - BlueField Management in NIC Mode

Contents:

12.1.1 Management Architecture

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



12.1.2 Management Interfaces

See this page for a detailed description of the interfaces of BlueField-3.

See this page for a detailed description of the interfaces of BlueField-2.

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

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

12.1.3 Recommended Management Approach

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

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

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

12.1.4 Management Methods

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

12.1.4.1 BlueField Update and Recovery

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

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

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

12.1.4.2 BlueField Monitoring

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

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

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

12.1.4.3 BlueField and Reset Control

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

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

12.2 Appendix - BlueField DHCP Discover

12.2.1 Possible Vendor Classes

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

12.2.2 BlueField DHCP DISCOVER

Sent from the BlueField UEFI BIOS to the DCHP server:

```
0.0.0.0 -> 255.255.255.255 DHCP 370 DHCP Discover - Transaction ID 0xfd2f32f1
192.168.254.119 -> 255.255.255.255 DHCP 342 DHCP Offer    - Transaction ID 0xfd2f32f1
        0.0.0.0 -> 255.255.255.255 DHCP 382 DHCP Request  - Transaction ID 0xfd2f32f1
192.168.254.119 -> 255.255.255.255 DHCP 342 DHCP ACK      - Transaction ID 0xfd2f32f1
aa:bb:cc:dd:ee:ff -> Broadcast    ARP 60 Who has 192.168.254.119?  Tell 192.168.254.139
gg:hh:ii:jj:kk:ll -> aa:bb:cc:dd:ee:ff ARP 60 192.168.254.119 is at gg:hh:ii:jj:kk:ll
  192.168.254.139 -> 192.168.254.119 TFTP 98 Read Request, File: /grubaa64.efi, Transfer type: octet,
tsize\000=0\000, blksize\000=1400\000, windowsize\000=4\000
  192.168.254.119 -> 192.168.254.139 TFTP 84 Option Acknowledgement, tsize\000=2414472\000,
blksize\000=1400\000, windowsize\000=4\000
  192.168.254.139 -> 192.168.254.119 TFTP 72 Error Code, Code: Option negotiation failed, Message: User
aborted the transfer
  192.168.254.139 -> 192.168.254.119 TFTP 90 Read Request, File: /grubaa64.efi, Transfer type: octet,
blksize\000=1400\000, windowsize\000=4\000
  192.168.254.119 -> 192.168.254.139 TFTP 70 Option Acknowledgement, blksize\000=1400\000,
windowsize\000=4\000
  192.168.254.139 -> 192.168.254.119 TFTP 60 Acknowledgement, Block: 0
  192.168.254.119 -> 192.168.254.139 TFTP 1446 Data Packet, Block: 1
  192.168.254.119 -> 192.168.254.139 TFTP 1446 Data Packet, Block: 2
```

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

Ethernet II, Src: aa:bb:cc:dd:ee:ff (aa:bb:cc:dd:ee:ff), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
    Destination: Broadcast (ff:ff:ff:ff:ff:ff)
        Address: Broadcast (ff:ff:ff:ff:ff:ff)
        .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default)
        .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
    Source: aa:bb:cc:dd:ee:ff (aa:bb:cc:dd:ee:ff)
        Address: aa:bb:cc:dd:ee:ff (aa:bb:cc:dd:ee:ff)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IP (0x0800)
Internet Protocol Version 4, Src: 0.0.0.0 (0.0.0.0), Dst: 255.255.255.255 (255.255.255.255)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport) (0x00)
    Total Length: 368
    Identification: 0x61c7 (25031)
    Flags: 0x00
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..0. .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0x17b7 [validation disabled]
        [Good: False]
        [Bad: False]
    Source: 0.0.0.0 (0.0.0.0)
    Destination: 255.255.255.255 (255.255.255.255)
User Datagram Protocol, Src Port: bootpc (68), Dst Port: bootps (67)
    Source port: bootpc (68)
    Destination port: bootps (67)
    Length: 348
    Checksum: 0xe557 [validation disabled]
        [Good Checksum: False]
        [Bad Checksum: False]
Bootstrap Protocol
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
   Hops: 0
    Transaction ID: 0xfd2f32f1
    Seconds elapsed: 0
    Bootp flags: 0x8000 (Broadcast)
        1... .... .... .... = Broadcast flag: Broadcast
        .000 0000 0000 0000 = Reserved flags: 0x0000
    Client IP address: 0.0.0.0 (0.0.0.0)
    Your (client) IP address: 0.0.0.0 (0.0.0.0)
    Next server IP address: 0.0.0.0 (0.0.0.0)
    Relay agent IP address: 0.0.0.0 (0.0.0.0)
    Client MAC address: aa:bb:cc:dd:ee:ff (aa:bb:cc:dd:ee:ff)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
    Option: (53) DHCP Message Type
        Length: 1
        DHCP: Request (3)
    Option: (54) DHCP Server Identifier
        Length: 4
        DHCP Server Identifier: 192.168.254.119 (192.168.254.119)
    Option: (50) Requested IP Address
        Length: 4
        Requested IP Address: 192.168.254.139 (192.168.254.139)
    Option: (57) Maximum DHCP Message Size
        Length: 2
        Maximum DHCP Message Size: 65280
    Option: (55) Parameter Request List
        Length: 35
        Parameter Request List Item: (1) Subnet Mask
        Parameter Request List Item: (2) Time Offset
        Parameter Request List Item: (3) Router
        Parameter Request List Item: (4) Time Server
        Parameter Request List Item: (5) Name Server
        Parameter Request List Item: (6) Domain Name Server
        Parameter Request List Item: (12) Host Name
        Parameter Request List Item: (13) Boot File Size
        Parameter Request List Item: (15) Domain Name
        Parameter Request List Item: (17) Root Path
        Parameter Request List Item: (18) Extensions Path
        Parameter Request List Item: (22) Maximum Datagram Reassembly Size
        Parameter Request List Item: (23) Default IP Time-to-Live
        Parameter Request List Item: (28) Broadcast Address
        Parameter Request List Item: (40) Network Information Service Domain
        Parameter Request List Item: (41) Network Information Service Servers
        Parameter Request List Item: (42) Network Time Protocol Servers
        Parameter Request List Item: (43) Vendor-Specific Information
        Parameter Request List Item: (50) Requested IP Address
        Parameter Request List Item: (51) IP Address Lease Time
        Parameter Request List Item: (54) DHCP Server Identifier
        Parameter Request List Item: (58) Renewal Time Value
        Parameter Request List Item: (59) Rebinding Time Value
        Parameter Request List Item: (60) Vendor class identifier
        Parameter Request List Item: (66) TFTP Server Name
        Parameter Request List Item: (67) Bootfile name
        Parameter Request List Item: (97) UUID/GUID-based Client Identifier
        Parameter Request List Item: (128) DOCSIS full security server IP [TODO]
        Parameter Request List Item: (129) PXE - undefined (vendor specific)
        Parameter Request List Item: (130) PXE - undefined (vendor specific)
        Parameter Request List Item: (131) PXE - undefined (vendor specific)
```

```
118             Parameter Request List Item: (132) PXE - undefined (vendor specific)
119             Parameter Request List Item: (133) PXE - undefined (vendor specific)
120             Parameter Request List Item: (134) PXE - undefined (vendor specific)
121             Parameter Request List Item: (135) PXE - undefined (vendor specific)
122         Option: (97) UUID/GUID-based Client Identifier
123             Length: 17
124             Client Identifier (UUID): c24e2ad6-a730-ec11-8000-08c0eb58d8ec
125         Option: (94) Client Network Device Interface
126             Length: 3
127             Major Version: 3
128             Minor Version: 16
129         Option: (93) Client System Architecture
130             Length: 2
131             Client System Architecture: Unknown (11)
132         Option: (60) Vendor class identifier
133             Length: 13
134             Vendor class identifier: NVIDIA/BF/PXE
135         Option: (255) End
136             Option End: 255
```

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

12.3 Appendix - Relating BlueField to Host and Port

To associate BlueField with its hosting machine for asset management, follow this procedure:

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

```
curl -k -u root:<password> -H 'Content-Type: application/json' -X GET https://<BF-BMC-IP>/redfish/v1/
Managers/Bluefield_BMC/EthernetInterfaces/eth0
 {
   "@odata.id": "/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0",
   "@odata.type": "#EthernetInterface.v1_6_0.EthernetInterface",
   "DHCPv4": {
     "DHCPEnabled": true,
     "UseDNSServers": true,
     "UseDomainName": true,
     "UseNTPServers": true
   },
   "DHCPv6": {
     "OperatingMode": "Stateful",
     "UseDNSServers": true,
     "UseDomainName": true,
     "UseNTPServers": true
   },
   "Description": "Management Network Interface",
   "FQDN": "dpu-bmc",
   "HostName": "dpu-bmc",
   "IPv4Addresses": [
     {
       "Address": "10.237.40.179",
       "AddressOrigin": "DHCP",
       "Gateway": "0.0.0.0",
       "SubnetMask": "255.255.0.0"
     }
   ],
   "IPv4StaticAddresses": [],
   "IPv6AddressPolicyTable": [],
   "IPv6Addresses": [
     {
       "Address": "fdfd:fdfd:10:237:966d:aeff:fe17:9f5f",
       "AddressOrigin": "DHCPv6",
       "AddressState": null,
       "PrefixLength": 64
     },
     {
       "Address": "fe80::966d:aeff:fe17:9f5f",
       "AddressOrigin": "LinkLocal",
       "AddressState": null,
       "PrefixLength": 64
     }
   ],
   "IPv6DefaultGateway": "fe80::445b:ed80:5f97:8900",
   "IPv6StaticAddresses": [],
   "Id": "eth0",
   "InterfaceEnabled": true,
   "LinkStatus": "LinkUp",
   "MACAddress": "94:6d:ae:17:9f:5f",
   "MTUSize": 1500,
   "Name": "Manager Ethernet Interface",
   "NameServers": [
     "fdfd:fdfd:7:77:250:56ff:fe8b:e4f9"
   ],
   "SpeedMbps": 0,
   "StaticNameServers": [],
   "Status": {
     "Health": "OK",
     "HealthRollup": "OK",
     "State": "Enabled"
```

```
    },
    "VLANs": {
      "@odata.id": "/redfish/v1/Managers/Bluefield_BMC/EthernetInterfaces/eth0/VLANs"
    }
  }
```

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

```
curl -k -u root:<password> -H "Content-Type: application/octet-stream" -X GET https://<bmc_ip>/redfish/v1/
Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/NetworkDeviceFunctions/eth0f0
{
  "@odata.id": "/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/NetworkDeviceFunctions/
eth0f0",
  "@odata.type": "#NetworkDeviceFunction.v1_9_0.NetworkDeviceFunction",
  "Ethernet": {
    "MACAddress": "02:b1:b6:12:39:05",
    "MTUSize": 1500
  },
  "Id": "eth0f0",
  "Links": {
    "OffloadSystem": {
      "@odata.id": "/redfish/v1/Systems/Bluefield"
    },
    "PhysicalPortAssignment": {
      "@odata.id": "/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapter/Ports/eth0"
    }
  },
  "Name": "NetworkDeviceFunction",
  "NetDevFuncCapabilities": [
    "Ethernet"
  ],
  "NetDevFuncType": "Ethernet"
}
```

12.4 Appendix - BMC and ERoT Upgrade Process for BlueField-2

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

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

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

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

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

BMC firmware update takes 15-20 mins.

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

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

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

```
TaskState="Completed"
TaskStatus="OK"
```

```
TaskProgress="100"
```

4. Reset/reboot the BMC.

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

5. To query the installed version, run:

   ```
   python3 openbmctool.py -H <bmc_ip> -U root -P <password> firmware running_version
   ```

   > The BlueField-2 BMC takes about 2.5 minutes to complete boot.

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

6. Trigger eROT update.

   ```
   python3 openbmctool.py -H <bmc_ip> -U root -P <password> apfirmware flash cec -f <path-to-signed-CEC-OTA-image-file>
   ```

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

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

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

   ```
   sudo ipmitool power cycle
   ```

12.5 Appendix - Software Upgrade Provisioning Flow

This appendix details the steps for provisioning software components on NVIDIA® BlueField®-3 networking platform (DPU or SuperNIC).

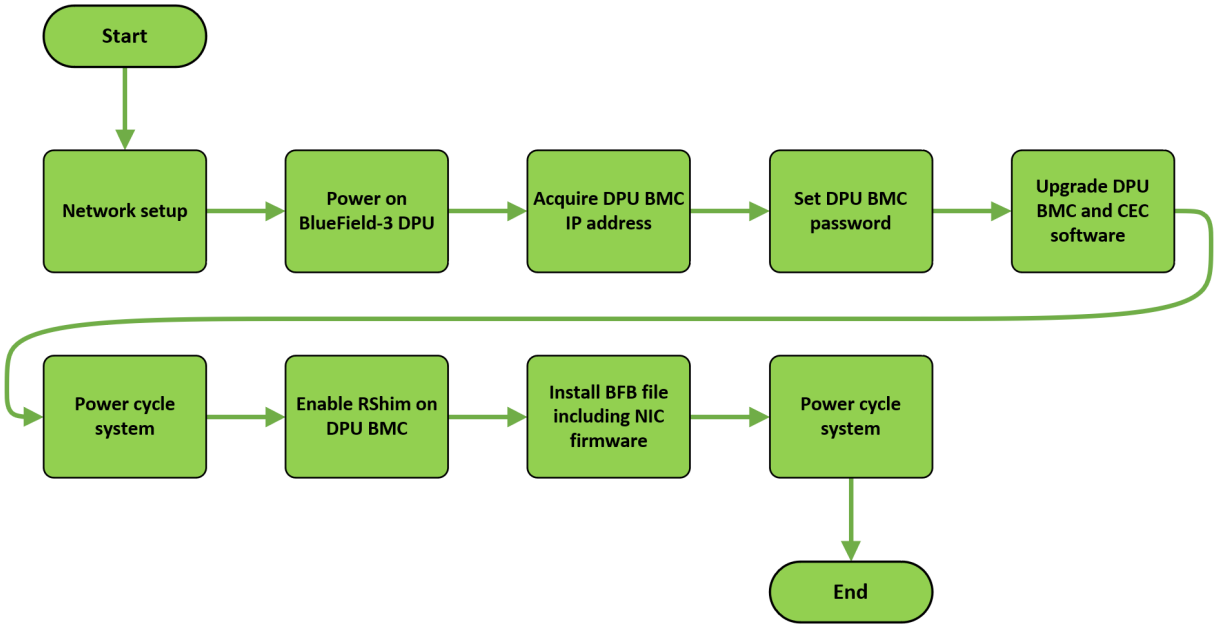
> The procedure for BlueField BMC software upgrade is agnostic to the version of the software. Once upgraded, however, the procedure assumes you to be running the latest BMC software.

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

- BlueField 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 device. 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/ComputerSystem.Reset
      ```

      Replace the parameters with the information relevant for your host BMC.
3. Acquire the BlueField BMC's MAC address from the label affixed to the BlueField (highlighted in the image). Use the BlueField BMC's MAC address to retrieve the assigned IP address from

the DHCP server to enable communication with the BlueField BMC over the network.

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

    ```
    curl -k -u root:<password> -X POST "https://<host_bmc_ip>/redfish/v1/Systems/1/Actions/
    ComputerSystem.Reset" -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 BlueField 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.

12.6 Appendix - Generic IPMI 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`
- `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>`

12.7 Appendix - NVIDIA OEM IPMI 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
`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 NVIDIA® BlueField® 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 BlueField from BMC under the OEM NetFn group `0x30`. • `0x00` - hard reset of BlueField
`0x32`	`0xa7`	N/A	Gets NTP servers.
`0x32`	`0xa8 0x00`	`0x31 0x2e 0x69 0x6e 0x2e 0x70 0x6f 0x6f 0x6c 0x2e 0x6e 0x74 0x70 0x2e 0x6f 0x72 0x67`	Adds primary NTP server with address `1.in.pool.ntp.org`
`0x32`	`0xa8 0x01`	`0x31 0x2e 0x69 0x6e 0x2e 0x70 0x6f 0x6f 0x6c 0x2e 0x6e 0x74 0x70 0x2e 0x6f 0x72 0x67`	Adds secondary NTP server with address `1.in.pool.ntp.org` ⚠ Should be added after a primary server has been declared.

netfunc	cmd	data	Description
`0x32`	`0xa8` `0x02`	`0x01`	Enable time sync to NTP server
`0x32`	`0xa8` `0x02`	`0x00`	Disables NTP time sync
`0x32`	`0xD2`	N/A	Triggers a CEC self reset for CEC pending firmware activation

13 Document Revision History

13.1 Rev 26.01 – February 16, 2026

Added:

- Section "Host Privileges Configuration"
- Section "Setting Host Privilege Configuration"

Updated:

- Section "Storage Health Monitoring"
- Section "Redfish NIC Subsystem Management"
- Section "IPMItool NIC Subsystem Management"

13.2 Rev 25.10 – November 06, 2025

Added:

- Page "Storage Health Monitoring"

Updated:

- Section "Boot Source Override"
- Page "Deploying BlueField Software Using BFB"
- Get the firmware inventory collection in "System Inventory"
- Section "Enabling Serial Redirect Mode to Run from Arm or BMC OS"
- Section "Enabling RShim on BlueField BMC"

13.3 Rev 25.07 – August 05, 2025

Added:

- Page "System Processor"

Updated:

- Section "ATX Power Error"
- Section "Boot Source Override Config Using IPMI"
- Section "BFB Live Firmware Update" in Installation for DPU Mode

13.4 Rev 25.04 – April 30, 2025

Updated:

- Section "Data Port Module High Power Consumption Notification"
- Section "Data Port Module Thermal 'Going High' Notification"
- Section "Data Port Module Thermal 'Going Low' Notification"
- Section "Reset Control Using IPMI"
- Section "SDR Sensor List"
- Section "Getting List of Support Sensors"

Added:

- Section "Voltage Sensors"
- Section "Power Sensors"
- Section "Power Deviation Sensors"
- Section "PCIe Correctable Error"
- Section "PCIe Uncorrectable Error"
- Section "PCIe Fatal Error"
- Section "ATX Power Error"
- Section "Getting Measurements"
- Section "RAS PCIe Error"
- Page "DPU BMC SPDM Attestation via Redfish"
- Section "Deleting Static IPv4 Address"
- Section "Adding IPv4 Address"
- Section "Root CA Management Commands"

13.5 Rev 25.01 – February 25, 2025

Updates to hierarchy under BlueField Management and BMC Management sections.

13.6 Rev 25.01 – February 6, 2025

Added:

- Page "Appendix - BlueField Management in NIC Mode"

Updated:

- Page "DPU Mode Installation"
- Page "BlueField Modes of Operation Configuration"
- Page "Deploying Software Using BFB"

13.7 Rev 24.10-LTSU1 FUR – January 02, 2025

Added:

- Page "Unsupported BMC Functionalities in NIC Mode"

13.8 Rev 24.10-LTSU1 – December 06, 2024

Added:

- Section "RAS Logging"

Updated:

- Section "Enabling Serial Redirect Mode to Run from Arm or BMC OS"
- Section "Getting 3-port Switch Ports Mode"
- Section "Configuring Router IPv6 Mode"
- Section "Configuring IPv6 Static Router IP"

- Section "Configuring IPv6 Static Router MAC"

13.9 Rev 24.10 – October 31, 2024

Added:

- Section "BIOS Debug Mode"
- Sensor `ddr_temp` to section "Temperature Sensors"
- Sensor `ddr_temp` to section "SDR Sensor List"
- Informational note to page "BlueField Arm State"
- Section "Data Port Module Events"
- Section "Arm Frequency Change Redfish System Command"
- Section "Data Port Module Temperature Going High"
- Section "Data Port Module Temperature Going Low"

Updated:

- Section "Enabling/Disabling IPv6 DHCP"
- Section "Serial Over LAN"
- Section "Arm Frequency Change"
- Section "BMC FRUs"
- Notes about the size of a single BMC on "BMC and BlueField Logs"

13.10 Rev 24.07 – August 14, 2024

Added:

- Page "Guest Tunnel"
- Page "Rsyslog"
- Section "Initial Provisioning of Golden Image to BMC"
- Section "Retrieving Golden Image Version Information"
- Section "Retrieving Golden Image Version Information Using Redfish"

Updated:

- Section "Redfish Management Interface"
- Diagram on page "CEC and BMC Firmware Operations"
- Multipart update with `MultipartHttpPushUri` command in section "Trigger Secure Firmware Update"
- Section "ForceUpdate"
- Section "Updating BMC"
- Section "Updating CEC"
- Section "Installing BFB"
- Section "Changing Default UEFI Password Using Redfish"
- Section "Retrieve Information on Pending Settings"
- Section "FRU Reading IPMI Commands"
- Section "Hard Rebooting BlueField"
- Section "Updating BMC"
- Section "RAS Errors"
- Section "System Commands"

- Page "BlueField Host Network Interface"

13.11 Rev 24.04 – May 05, 2024

Added:

- Page "BlueField Host Network Interface"
- Section "LLDP in Redfish"
- Page "BlueField Arm State"
- Section "Product Instance Identifier"
- Section "CEC Activation and Reset" under "BMC and CEC Firmware Update"
- Section "BMC DPU Information"

Updated:

- Section "Installing BFB"
- Section "FRU Reading IPMI Commands"

13.12 Rev 24.01 – February 08, 2024

Added:

- Section "Monitoring DPU OS Shutdown from BMC"
- Section "Example for CSR Generation, Certificate Creation and Replacement"
- Page "BlueField Modes of Operation Configuration"
- Section "Redfish Event Log"
- Cmds `0xa8 0x00` and `0xa8 0x01` to "NVIDIA OEM Commands"

Updated:

- Section "Changing Default Password"
- Section "BIOS CA Certificates"
- Section "System Commands"
- Section "SEL Redfish Commands"
- Section "Get Maximum Power Capacity"
- Section "Get Dump Task State"
- Section "BMC and CEC Firmware Update"

13.13 Rev 23.10 – 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 "Factory Reset BMC"
- Section "Reset or Reboot BMC"
- Section "BMC Sensor Data"
- Section "Serial Over LAN"
- Section "BMC Dump Operations"

13.14 Rev 23.09 – September 20, 2023

Added:

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

Updated:

- Section "System FRU"
- Section "System Logs"
- Section "List of IPMI Supported FRUs"
- Section "Boot Configuration"
- Section "BIOS Secure Boot Configuration"
- Section "BIOS Configuration"
- Section "Reset Control"

13.15 Rev 23.07 – 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 Configuration"
- Section "Installing BFB"
- Section "BMC and CEC Firmware Update" and its subsections

13.16 Rev 23.04 – 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"

13.17 Rev 2.8.2-34 – October 21, 2022

Added:

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

Updated:

- Section "Boot Configuration" with note on DPU boot override setting

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. Neither NVIDIA Corporation nor any of its direct or indirect subsidiaries and affiliates (collectively: “NVIDIA”) make any representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice. Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

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

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

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

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

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

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA