BlueField Management

Table of contents

Management in BlueField BMC

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

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

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

Managing the BlueField using its BMC is detailed hereafter.

Remote Management Using Redfish Protocol

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

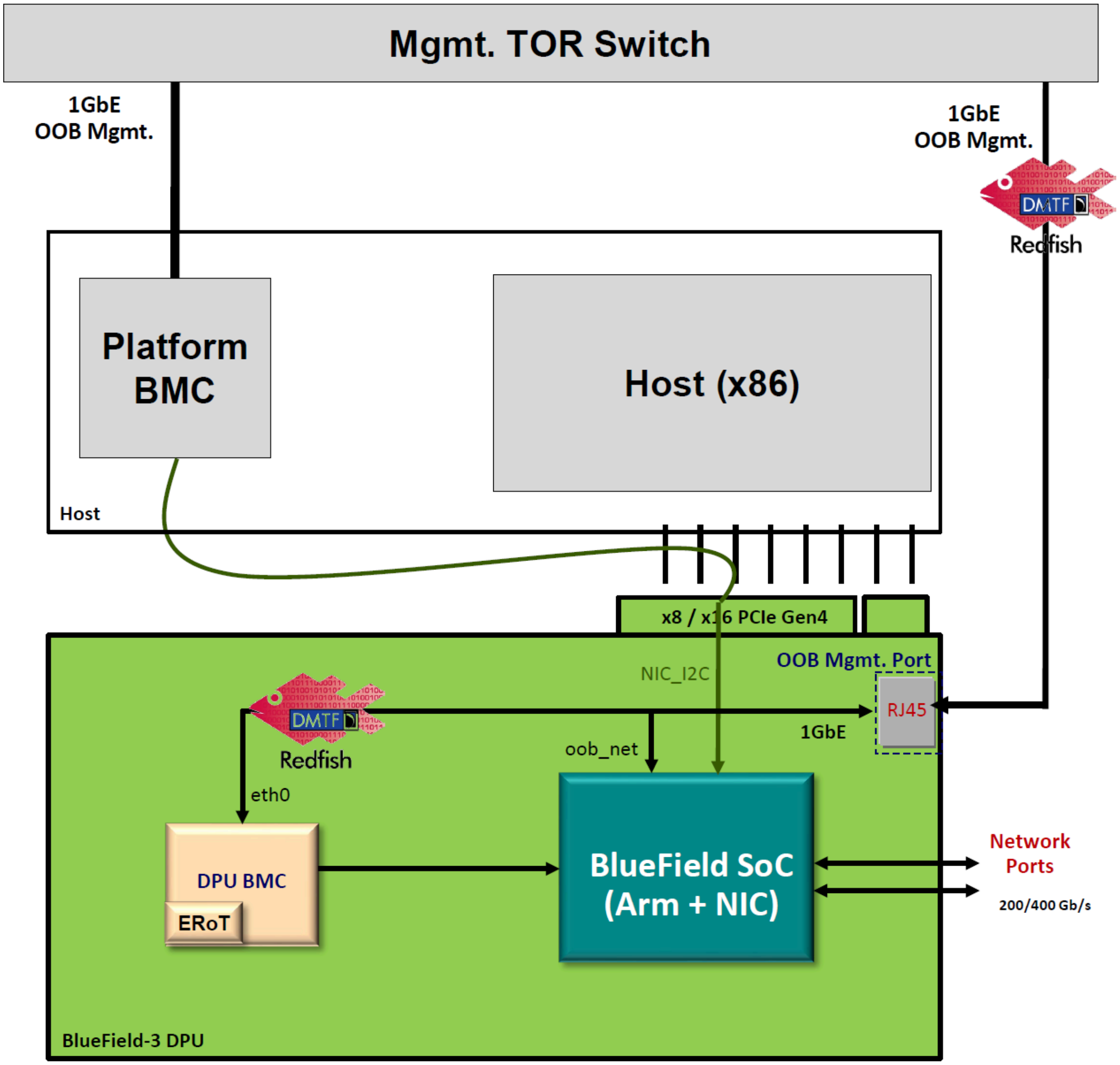
Redfish is supported from the host BMC and BlueField BMC.

Redfish replaces IPMI, providing the following advantages:

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

Management Architecture

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



Management Interfaces

Info

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

Info

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

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

Management Interface	Description	Comment
OOB Management Port (1GbE RJ45)	A dedicated, separate Ethernet interface to manage the BlueField from the remote management controller (RMC)	**Info** 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`). **Note** IPMI is supported for backward compatibility, but it is recommended to start new deployments with Redfish only.
SMBus (PCIe Golden	Enables PLDM/NC-SI over MCTP between	Enables the host BMC to monitor the BlueField

Management Interface	Description	Comment
Fingers)	the BlueField and the host BMC	
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 ⓘ **Info** Unavailable while in zero-trust mode. Use the 1GbE OOB interface instead.

Recommended Management Approach

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

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

Management Methods

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

ⓘ **Note**

The following method are not supported by BlueField SuperNIC.

BlueField Update and Recovery

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

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

Each file format has a different update and recovery method:

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

ⓘ **Note**

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

BlueField BMC Update

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

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

ⓘ **Info**

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

ⓘ **Note**

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

BlueField Monitoring and Telemetry

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

- BlueField temperatures (board, DDR, and ports), voltages and link states

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

Info

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

BlueField and BlueField BMC Reset Control

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

Info

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

BlueField UEFI Configuration

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

Info

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

Console Interface

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

ⓘ **Info**

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

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

Common Configurations

This section contains the following pages:

- BlueField Modes of Operation Configuration
- BIOS Secure Boot Configuration
- BIOS Configuration

BlueField Modes of Operation Configuration

Getting Mode of Operation

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

Info

Current setting appears under `Mode`.

Setting DPU Mode

1. Run the following Redfish commands:
   - For NVIDIA® BlueField®-3:

```
curl -k -u root:'<PASSWORD>' -H "Content-Type:
application/json" -X POST -d '{"Mode":"DpuMode"}'
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Oem/Nvidia/A
```

  - For BlueField-2:

```
curl -k -u root:'' -H "Content-Type: application/json" -
X PATCH -d '{"Attributes":{"NicMode":"DpuMode"}}'
https:///redfish/v1/Systems/Bluefield/Bios/Settings
```

2. Reboot the BlueField Arm cores twice.
  - To reboot Arm cores from the Arm OS:

```
reboot
```

  - To reboot Arm cores from the remote host:

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

3. Power cycle BlueField by running the following command from the remote host :

```
ipmitool chassis power cycle
```

Setting NIC Mode

1. Run the following Redfish commands:
  - For BlueField-3:

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

- For BlueField-2:

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

2. Reboot the BlueField Arm cores twice.
    - To reboot Arm cores from the Arm OS:

    ```
    reboot
    ```

    - To reboot Arm cores from the remote host:

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

3. Power cycle BlueField by running the following command from the remote host :

```
ipmitool chassis power cycle
```

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.

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"
}
```

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/Bluefiel
d'{"SecureBootEnable":true}'
```

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

The included page could not be found.

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/Secur
  ```

  Output example:

  ```
  {
    "@odata.id":
  "/redfish/v1/Systems/Bluefield/SecureBoot/SecureBootDatabases"
    "@odata.type":
  "#SecureBootDatabaseCollection.SecureBootDatabaseCollection",
    "Members": [
      {
  ```

```
      "@odata.id":
"/redfish/v1/Systems/Bluefield/SecureBoot/SecureBootDatabases/
    },
    ..
    {
      "@odata.id":
"/redfish/v1/Systems/Bluefield/SecureBoot/SecureBootDatabases/
    },
..
    {
      "@odata.id":
"/redfish/v1/Systems/Bluefield/SecureBoot/SecureBootDatabases/
    },
..
..
    {
      "@odata.id":
"/redfish/v1/Systems/Bluefield/SecureBoot/SecureBootDatabases/
    },
..
  ],
  "Members@odata.count": 10,
  "Name": "UEFI SecureBoot Database Collection"
}
```

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

Note

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/IwDQYJKoZI
BQAwVjELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAk5DMRAwDgYDVQQHEwdS
MQ8wDQYDVQQKEwZMZW5vdm8xFzAVBgNVBAMTDkxlbm92byBVRUZJIERC
MDMxNTIxMTYzNFoXDTQxMDMxNTIxMTYzNFowVjELMAkGA1UEBhMCVVMx
BAgTAk5DMRAwDgYDVQQHEwdSYWxlaWdoMQ8wDQYDVQQKEwZMZW5vdm8x
BAMTDkxlbm92byBVRUZJIERCMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A
AQEA1ezmdoBlk5yDVuXw8F774CvllhMz9bT0/TpH3kmRpPAizSRDzbHL
W4zjdLxTqOlwZt6UUcWxlyzKcoDPe43cE6YH1kM/rscvm3AaVL+4GcyG
QFHWER25xCTokMsCdKB42Ty7hWW5FBPepgAS+GDfqQfb/4hoonIlen5X
RM1DIVBUiIbJdgERYeoGjY/Rh4A1VWl6ErzyzokYnf63JjSFR2kVV0ap
7qBd1RNHwQrCAryRADX1XGvRZURzwQdEXfOqZOkVjNKr1fD761qvPE8T
mciMocIXqoqWKPAkgbMwKmcsFQIDAQABozMwMTAPBgNVHRMBAf8EBTAD
A1UdDwEB/wQEAwICBDAOBgNVHQ8Baf8EBAMCB4AwDQYJKoZIhvcNAQEL
AJ2U0UjB+sxF/HE5sY56vJbdFITl8o0Yf7XJImL0VtgpYjfeqiE768G2
hDOps3+4w4p8FUSO6StzCz6UuUyxlUjQzpkxZ97Ouq1sGhjy7dZybTEB
l1EpJSfBiwxTdm7svJoABKs8Hs7e9f3XX5PK76SxllMbDaxAm7UvCppY
gWt3rGRiO3W6pfd07ioCD03kgGzYNOZeU2S+maE1Xt4kUoYs3HxyrhJG
4w5LfCKrlxi+3KMf+vXxEBfGYBvjwcA7KCW92GnUQGVjZbEGs6EaTBx7
oWS/500qiwNRp2xqdBxg1d0=
-----END CERTIFICATE-----
```

```
curl -k -u root:'<password>' -H 'Content-Type:
application/json' -X POST
https://<bmc_ip>/redfish/v1/Systems/Bluefield/SecureBoot/Secur
-d\
 '{"CertificateString": "-----BEGIN CERTIFICATE-----
\nMIIDbTCCAlWgAwIBAgIUO2MdJt2cTCGr0eO4PiBV5Uk0b/IwDQYJKoZIhvcN
----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/Secur
-d\
'{"SignatureString":
"80B4D96931BF0D02FD91A61E19D14F1DA452E66DB2408CA8604D411F92659
"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/Secur
```

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/Secur
-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"
}
```

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/Secur
-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"
}
```

Tip

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/
  },
  "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/Computer
-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"`

BIOS Configuration

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

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.

Getting BIOS Attributes List

Info

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/BiosA
```

Output example:

Info

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
}
```

Info

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.

Getting Current BIOS Attributes Value

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

Output example:

Info

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",
 ...
}
```

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.

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:

Info

`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"
}
```

Info

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

BIOS Configuration Examples

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":"NewPa
```

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.

BIOS CA Certificates

Viewing Currently Installed BIOS CA Certificates

⚠ **Warning**

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"
}
```

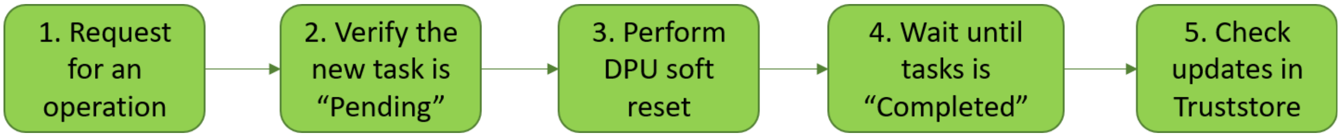
2. Trigger the following GET request to view the details of a specific certificate:

```
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"
}
```

BIOS CA Certificates Collection Operations


1. Request for an operation
2. Verify the new task is “Pending”
3. Perform DPU soft reset
4. Wait until tasks is “Completed”
5. Check updates in Truststore

1. Request for an operation:

    - To install a certificate, trigger the following POST request which contains the certificate string and type in JSON format:

Note

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.

Warning

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/T
-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`.

Note

The "\n" at the end are mandatory.

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

1.
    - 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/T
      ```

    - 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.Re
      -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/Computer
-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.

BIOS Attributes

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

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.

Update and Recovery

This section contains the following pages:

- System Inventory
- Deploying Software Using BFB
- Boot Configuration

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

> **Info**
>
> 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"
}
```

Note

- 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

Info

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/DP
```

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"
}
```

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.

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	• BF-Bundle BFB (firmware + Arm OS + DOCA) • BF-fwBundle BFB (firmware only) • Per-SKU BF-fwBundle BFB

Update Flow	Description	Supported Image Types
	involves downtime. This flow supports recovery as well.	
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.

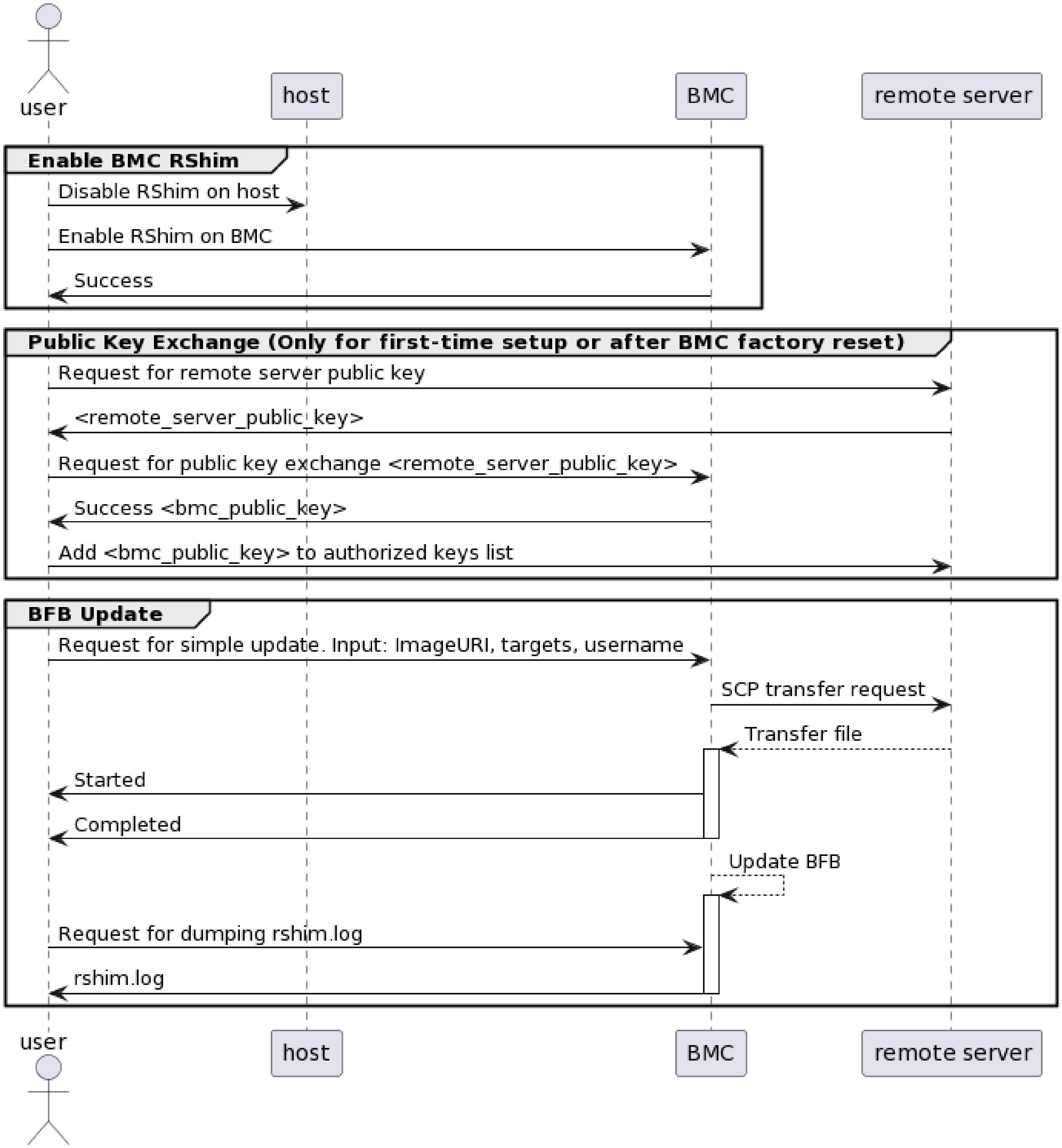
BFB Installation Procedure

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.

	Stage	Description
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).

Update Flow


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

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.

Update Flow Image Transfer

Offline Update Flow

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

Note

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

Info

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

Deferred Update Flow

Note

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.

Note

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
```

Note

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
```

Note

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"
}
```

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>`).

> **Note**
>
> 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`

> **Note**
>
> 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

Setting Up Secure Connection

Note

Relevant only for SCP users with Redfish.

Note

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

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

   Info

   OpenSSH is required for this step.

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

   Where:

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

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

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

   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"
        }
    ]
  }
```

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>
```

Note

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"
```

Installation Status and Activation

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/LogServ
   ```

   Where:

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

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

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

   Where:

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

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

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

```
https://<ip_address>/redfish/v1/Managers/Bluefield_BMC/LogServ
--output </path/to/tar/log_dump.tar.xz>
```

Where:

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

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

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

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

   Note

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

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

   ```
   curl -k -u root:"<PASSWORD>" -H "Content-Type:
   application/json" -X GET
   ```

```
https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory/DP
```

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"
```

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"
}
```

Note

The `UpdateStatus` can be:

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

  0000019b-22f7-dcdf-a9fb-b6f7c5430003

- `'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.

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	- DPU Mode	1. (DPU Mode only) Gracefully shut down the BlueField Arm OS.	- The firmware update is

Reset Type	Mode of Operation	Applying Reset Steps	Notes
Power Cycle)	- NIC Mode	2. Perform a full server power cycle.	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	- Relevant to Deferred Update Flow only. - The next warm reboot will: ◦ Gracefully shut down BlueField Arm cores

Reset Type	Mode of Operation	Applying Reset Steps	Notes
		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.	◦ Reset the NIC, Arm Complex, and BMC ◦ Boot from the new firmware image

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
```

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"
```

ⓘ **Info**

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

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

Where:

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

Then repeat the following steps:

1. Preparing Secure Access for Image Transfer.
2. 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"
```

Info

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

- If SCP fails:

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

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.

Boot Config Using Redfish

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
```

Info

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/<boo
option>
```

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/BootOpt
```

- 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/BootOpt
id>
```

Applying Pending Boot Configurations

Note

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/BootOpt
id> -d '{"BootOptionEnabled": false}'
```

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(0x0,0
}
```

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

Note

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

```
curl -k -u root:<password> -X PATCH -d '{ "Boot": {
"BootOrder": [ "Boot0006", "Boot0017", "Boot0001",
"Boot0002", "Boot0003", "Boot0004", "Boot0005", "Boot0007", ]
}}' https://<BF-BMC-
```

```
IP>/redfish/v1/Systems/<SystemID>/Settings | python3 -m
json.tool
```

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

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:

> **Info**
>
> 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"] }}'
```

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.

Boot Source Override Config Using Redfish

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"
    ...
   }
```

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"
  }
}'
```

Note

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

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/Boot
    ```

    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"
  }
}'
```

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.

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.

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
```

Note

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

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
```

Note

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

Note

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

Resetting Boot Override to Default

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

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

Setting Persistent PXE Boot

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

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

Info

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

Info

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

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.

Monitoring

This section contains the following pages:

- System FRU
- System Logs
- BMC Sensor Data
- BlueField Arm State
- Rsyslog
- DPU Chassis
- DPU Information
- BMC and BlueField Logs
- System Processor

System FRU

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",
...
```

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:

Note

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>
```

Note

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

Info

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

System Logs

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.

Event Log Redfish Commands

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/LogServ
    }
  },
  "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"
}
```

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/
```

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/attac
      "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"
}
```

Clearing Event Log

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

SEL Redfish Commands

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.
    }
  },
  "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"
 }
```

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/Entr
```

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"
}
```

Clearing Event Log

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

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/Nvi
-d '{"ErrorInfoCap":300 }'
```

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/SelCa
```

Example output:

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

SEL IPMI Commands

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

Displaying SEL Information

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

Displaying List of Events

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

Displaying Extended List of Events

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

Saving SEL Events to File

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

Clearing SEL

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

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]>
```

Getting SEL Info Log Capacity

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

```
0x4b
```

SEL Message Types

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

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
```

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
```

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
```

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
```

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
```

System Commands

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
```

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
```

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
```

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
```

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
```

RAS Errors

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
```

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
```

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
```

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
```

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
```

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
```

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
```

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
```

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
```

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
```

Info

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

Data Port Module Events

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
```

Info

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

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

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
```

Info

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

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

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
```

Info

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

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

Redfish Event Log

System Commands

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/atta
      "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"
    }
```

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/attac
      "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/attac
      "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/attac
      "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"
    }
```

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/atta
      "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"
    },
```

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/attac
      "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"
    },
```

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"
    }
```

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/atta
      "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"
    }
```

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/attac
      "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"
    }
```

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/attac
      "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"
    }
```

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/atta
      "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"
    }
```

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/atta
      "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"
    }
```

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/atta
      "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"
    }
```

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/attac
      "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"
    }
```

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/atta
      "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"
    },
```

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/atta
      "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"
    }
```

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/attac
      "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"
    },
```

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/attac
      "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"
    }
```

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/atta
      "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"
    }
```

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/atta
   "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"
  }
```

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/atta
      "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"
    }
```

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/atta
      "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"
    }
```

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/atta
      "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"
    }
```

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/atta
      "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"
    }
```

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/atta
      "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"
    }
```

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/atta
      "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"
    }
```

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/attac
      "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"
    }
```

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/attac
      "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"
    }
```

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/atta
      "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"
    }
```

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/attac
      "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"
    }
```

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/attac
      "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"
    }
```

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/atta
      "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"
    }
```

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/atta
      "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"
    }
```

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/atta
      "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"
    }
```

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/atta
      "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"
    }
```

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/attac
      "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"
    }
```

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/attac
      "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"
    }
```

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/attac
```

```
      "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"
    },
```

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/attac
      "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"
    },
```

Info

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

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/attac
      "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"
    }
```

Info

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

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/atta
  "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"
}
```

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/atta
  "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"
}
```

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/atta
      "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"
    }
```

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/attac
      "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"
    }
```

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/atta
      "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"
    }
```

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/atta
      "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"
    }
```

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/atta
      "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"
    }
```

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/at
      "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"
    }
```

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"
    }
```

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"
    }
```

RAS Logging

CPER to Redfish severity translation:

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

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/attac
   "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"
}
```

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/attac
    "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"
}
```

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/atta
    "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":
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
                        "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":
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
                    },
                    "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"
}
```

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/////wEAAQAAAAAAAAAD4AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    "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"
}
```

BMC Sensor Data

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 O • 0x200 – connection er
`p1_link`	Discrete	IPMB	Uplink port 1 link status • 0x100 – connection O • 0x200 – connection er
`bluefield_temp`	Temperature	IPMB	NVIDIA® BlueField® tempera
`p0_temp`	Temperature	IPMB	Uplink port 0 SFP temperat
`p1_temp`	Temperature	IPMB	Uplink port 1 SFP temperat
`ddr_temp`	Temperature	IPMB	DDR temperature
`rtc_voltage`	Voltage	IPMB	RTC battery voltage
`power_envelope`	Power	IPMB	• This sensor indicates t maximum power cons allowed for BlueField-3 • This sensor is incompa secured Linux

Sensor Name	Sensor Type	Source	Description
`soc_power`	Power	IPMB	• This sensor indicates t current power consum the BlueField-3 DPU • This sensor is incompa secured Linux
`power_envelope_deviation`	Power	Synthesized Sensor	Measures the deviation of `soc_power` sensor value f `power_envelope` sensor • `power_envelope_de = soc_power - power_envelope` • The sensor value shou negative for normal c If the sensor value is p then SoC power has e the allowed power env • If the value of `soc_p` `power_envelope` se NaN, then the `power_envelope_de` sensor value will also b
`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	
`3_3V_RGM`	Voltage	BMC ADC	
`5V`	Voltage	BMC ADC	
`12V_ATX`	Voltage	BMC ADC	Input power rail from ATX (from gold fingers in case of

Sensor Name	Sensor Type	Source	Description
			when ATX power is off)
`12V_PCIe`	Voltage	BMC ADC	Input power rail from gold fi
`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	

Note

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

Sensor Redfish Commands

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"
}
```

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
    }
  }
}
```

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"
    }
  ]
}
```

Sensor IPMI Commands

BMC software supports reading chassis sensor information using the IPMItool.

The following subsections list commands which allow reading SDR data.

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
```

Displaying Extended Sensor Data

Displays extended sensor information.

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

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
```

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>
```

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>
```

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>
```

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>
```

BlueField Arm State

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

Info

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

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`

Monitoring BlueField Arm State Using IPMI

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

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.

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"
```

Note

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

Rsyslog Servers Configurations

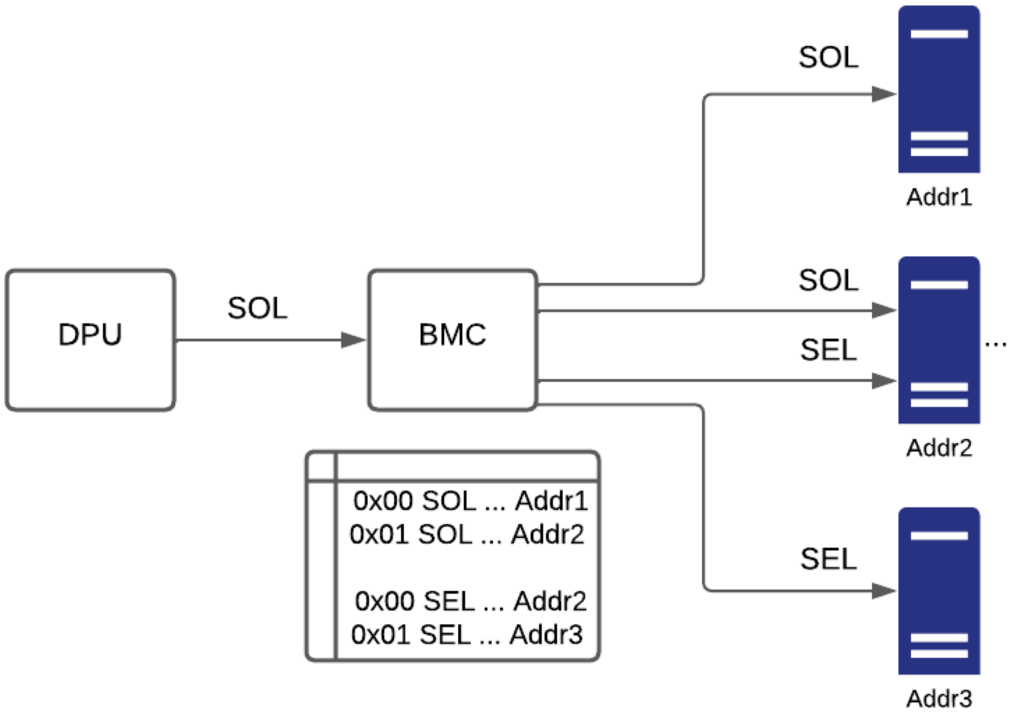
The rsyslog configurations define data streams. Each of them includes:

- 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

ⓘ **Note**

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.

IPMI Commands

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

<table>
<tr><th>netfunc</th><th>cmd</th><th>data</th><th>Description</th></tr>
<tr>
<td><code>0x32</code></td>
<td><code>0xD3</code></td>
<td><code><Index> <LogType></code></td>
<td>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:
<pre>
<Index> <LogType>
<Status>
<TransportProtocol>
<NetworkProtocol>
<ServerAddress> <Port>
</pre>
<ul>
<li>Byte 1 – Completion code:
<ul>
<li>0x00 – Success (does not appear in IPMI textual response)</li>
<li>0x01 – Failure (the rest does not appear in IPMI response)</li>
</ul></li>
<li>Byte 2 – Index
<ul>
<li>Index of server (0x00-0x09)</li>
</ul></li>
<li>Byte 3 – LogType
<ul>
<li>0x01 – SEL</li>
<li>0x03 – SOL</li>
</ul></li>
</ul>
</td>
</tr>
</table>

netfunc	cmd	data	Description

			• 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.

`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

netfunc	cmd	data	Description
			◦ 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.

Usage Examples

Setting Rsyslog Status of Two Configurations

The following commands create or modify two different rsyslog configurations with I ndex 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
```

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
```

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
```

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
```

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
```

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"
}
```

Chassis Card1

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

Output example:

```
{
  "@odata.id": "/redfish/v1/Chassis/Card1",
  "@odata.type": "#Chassis.v1_21_0.Chassis",
  "Actions": {
    "#Chassis.Reset": {
      "@Redfish.ActionInfo":
"/redfish/v1/Chassis/Card1/ResetActionInfo",
      "target": "/redfish/v1/Chassis/Card1/Actions/Chassis.Reset"
    }
  },
..
  "ChassisType": "Card",
  "EnvironmentMetrics": {
    "@odata.id": "/redfish/v1/Chassis/Card1/EnvironmentMetrics"
  },
  "Id": "Card1",
  "Links": {
    "ComputerSystems": [
      {
        "@odata.id": "/redfish/v1/Systems/Bluefield"
      }
    ],
    "Contains": [
      {
        "@odata.id": "/redfish/v1/Chassis/Bluefield_ERoT"
      },
      {
        "@odata.id": "/redfish/v1/Chassis/Bluefield_BMC"
      }
    ],
    "ManagedBy": [
      {
        "@odata.id": "/redfish/v1/Managers/Bluefield_BMC"
      }
    ]
  },
  "Manufacturer": "Nvidia",
  "Model": "Bluefield 3 SmartNIC Main Card",
  "Name": "Card1",
  "NetworkAdapters": {
    "@odata.id": "/redfish/v1/Chassis/Card1/NetworkAdapters"
  },
  "PCIeDevices": {
    "@odata.id": "/redfish/v1/Chassis/Card1/PCIeDevices"
  },
  "PCIeSlots": {
    "@odata.id": "/redfish/v1/Chassis/Card1/PCIeSlots"
  },
  "PartNumber": "900-9D3B4-00EN-EAB   ",
  "Power": {
    "@odata.id": "/redfish/v1/Chassis/Card1/Power"
  },
  "PowerState": "On",
  "PowerSubsystem": {
    "@odata.id": "/redfish/v1/Chassis/Card1/PowerSubsystem"
  },
  "SKU": "",
  "Sensors": {
    "@odata.id": "/redfish/v1/Chassis/Card1/Sensors"
  },
  "SerialNumber": "MT2245X00175            ",
  "Status": {
    "Conditions": [],
    "Health": "OK",
    "HealthRollup": "OK",
    "State": "Enabled"
  },
  "Thermal": {
    "@odata.id": "/redfish/v1/Chassis/Card1/Thermal"
  },
  "ThermalSubsystem": {
    "@odata.id": "/redfish/v1/Chassis/Card1/ThermalSubsystem"
  },
  "TrustedComponents": {
    "@odata.id": "/redfish/v1/Chassis/Card1/TrustedComponents"
  },
  "UUID": ""
}
```

Chassis Card1 NetworkAdapters

Note

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/NvidiaNe
```

Example output:

```
{
  "@odata.id":
"/redfish/v1/Chassis/Card1/NetworkAdapters/NvidiaNetworkAdapte
  "@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/NvidiaNe
```

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"
}
```

Note

Removing or adding new ports requires a BMC reboot.

DPU Information

Note

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

Getting Base GUID

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

Getting Base MAC

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

Getting Description

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

BMC and BlueField Logs

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

Two types of dumps are supported:

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

BMC Dump Operations

The following subsections list BMC dump operations.

Creating BMC Dump Task

Create a BMC dump task and gets the task ID.

Info

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/
```

Where:

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

Note

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

Note

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.

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

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/
--output </path/to/tar/log_dump.tar.xz>
```

Where:

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

Info

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`
- `procfd.log`

- `dreport.log`
- `disk-usage.log`
- `summary.log`

Note

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

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
```

Where:

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

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

System Dump Operations

The following subsections list system dump operations.

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
```

Where:

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

Info

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

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

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/
--output </path/to/tar/system_dump.tar.xz>
```

Where:

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

Info

After downloading, untar the file to view the logs.

Dump list:

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

  Note

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

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

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/
```

Where:

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

Info

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

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

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

BlueField Console Log

BMC captures the 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.

System Processor

Note

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.

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.

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`.

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"
}
```

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).
`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.

Network

This section contains the following pages:

- BlueField Host Network Interface
- OOB Network 3-Port Switch Control

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).

> **Note**
>
> These abilities are supported when operating in DPU mode only.

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"
}
```

Displaying Network Interface Object

Info

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`.

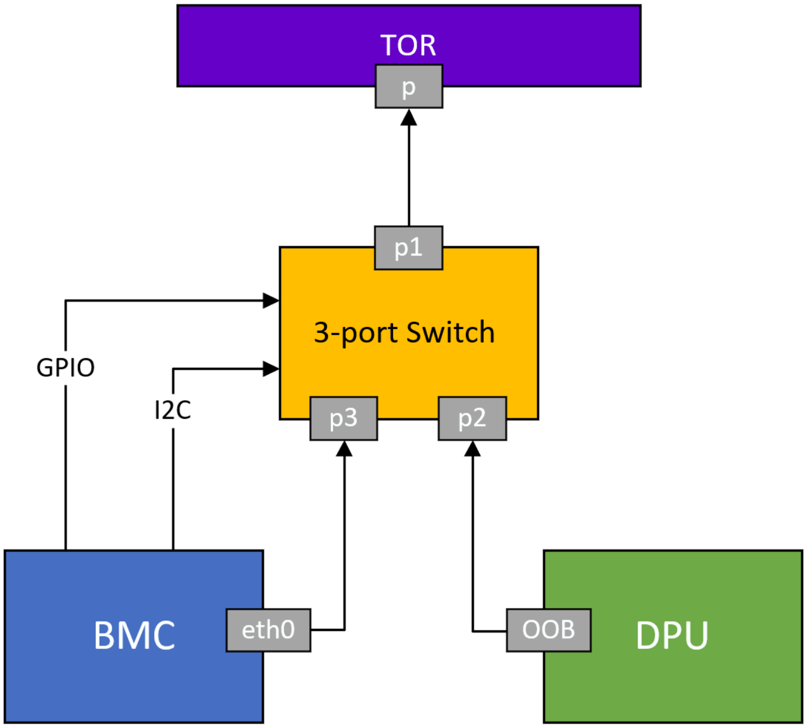
```
curl -k -u root:'<password>' -H 'Content-Type: application/json'
-X GET
https://<bmc_ip>/redfish/v1/Systems/Bluefield/EthernetInterfaces/o
{
  "@odata.id":
"/redfish/v1/Systems/Bluefield/EthernetInterfaces/oob0",
  "@odata.type": "#EthernetInterface.v1_6_0.EthernetInterface",
  "DHCPv4": {
    "DHCPEnabled": 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"
  }
}
```

Note

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

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.



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

netfunc	cmd	data	Description
`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

Info

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
```

3-Port Switch Redfish Commands

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

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

Note

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.

Note

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"
        }
    ]
}
```

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.Res
```

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"
        }
    ]
}
```

Miscellaneous

This section contains the following topics:

- Bare-metal Reprovisioning
- Power Capping
- RShim Over USB
- Serial Over LAN
- Serial Redirect Mode
- Vendor Field Mode

Bare-metal Reprovisioning

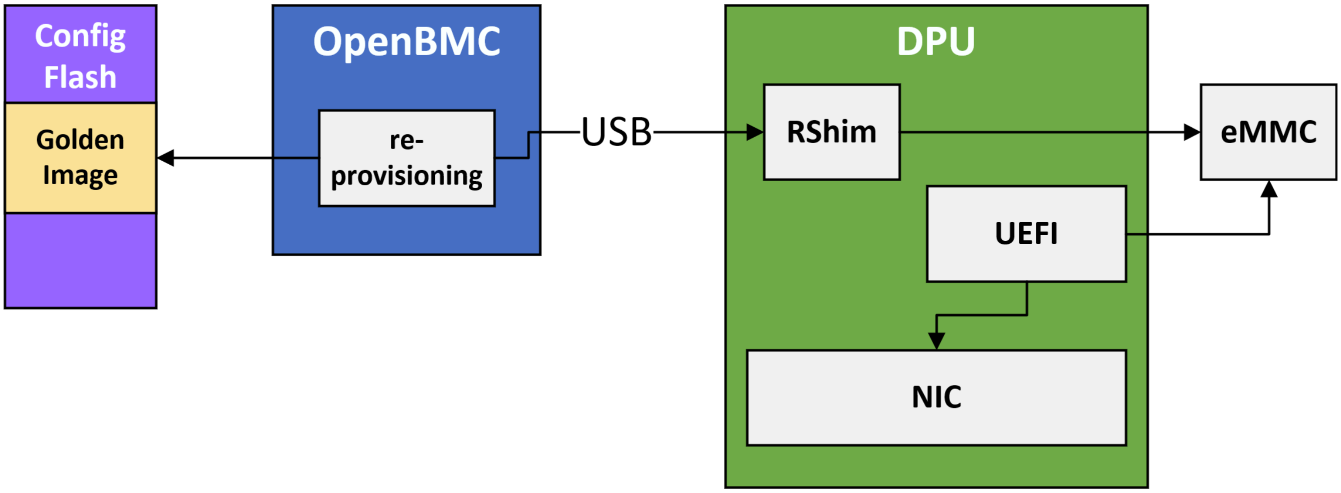
Note

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.

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/
```

> **Info**
>
> After the image is copied to the BMC's volatile memory, the version is extracted and stored to enable specific features.

> **Note**
>
> 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>
     ```

Info

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`.

Retrieving Golden Image Version Information

Note

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
  ```

Retrieving Golden Image Version Information Using Redfish

> **Note**
>
> 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'
```

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.SimpleU
      ```

    - 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.SimpleU
      ```

      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.

Info

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.SimpleUpdat
```

- 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.SimpleUpdat
```

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.

Note

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".

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.

  > **Note**
  >
  > 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

    > **Info**
    >
    > Reprovisioning messages are prefixed with `[<running date> GOLDEN-IMAGE-RECOVERY]`.

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.

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"
```

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.

Power Capping

Info

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.

Redfish Power Capping Requests

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"
  }
}
```

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">}'
```

Info

The ability to adjust power capping is disabled by default.

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`).

Note

If user configuration is lower than the minimum capacity power or higher than the maximum capacity power, then BMC will return error.

IPMI Power Capping Commands

Getting Power Capping Status

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

Enabling/Disabling Adjustment of Power Capping

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

Where `val`:

- 0 – disable
- 1 – enable

Note

Changeable only from BMC prompt using `admin` account.

Info

The ability to adjust power capping is disabled by default.

Getting Power Capping Percentage

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

Setting Power Capping Percentage

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

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

> **Note**
>
> Changeable only from BMC prompt using `admin` account.

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

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

Getting Maximum Power Capacity

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

> **Info**
>
> Power is given in watts.

Getting Minimum Power Capacity

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

Info

Power is given in watts.

RShim Over USB

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

Enabling RShim on BlueField BMC

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

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

Info

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
```

Note

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/Actio
```

0000019b-22f7-dcdf-a9fb-b6f792ef0000

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
```

Info

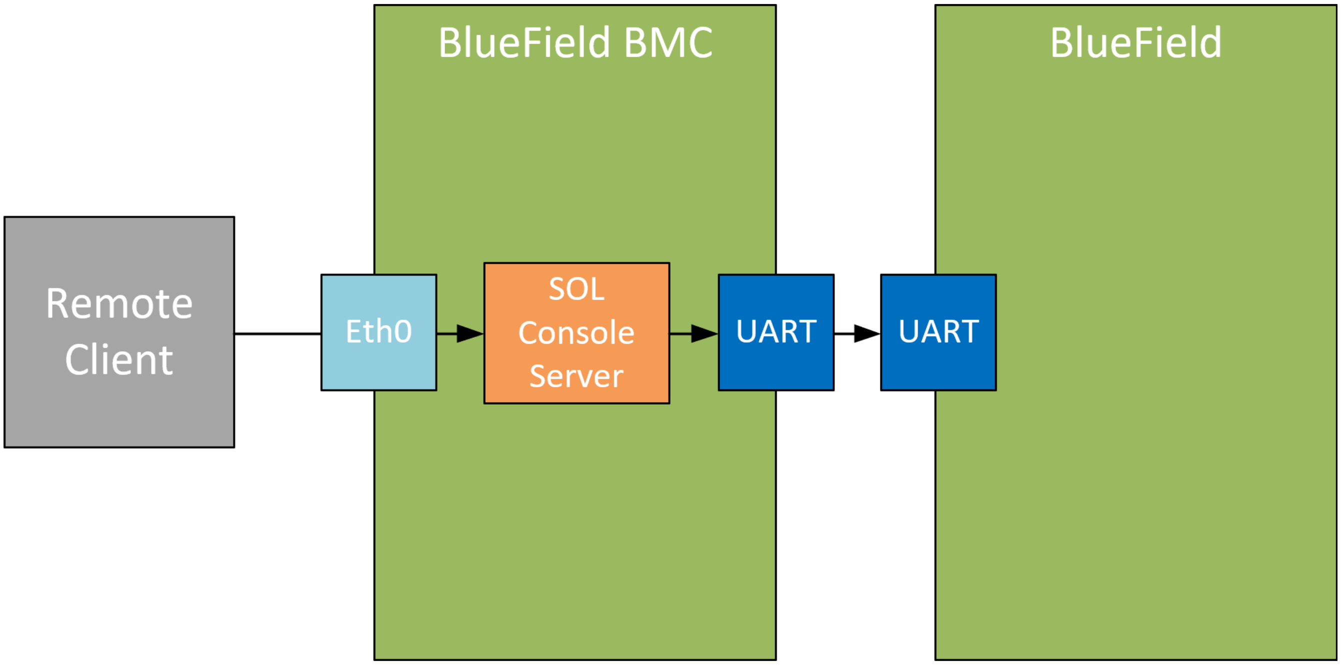
This may take up to 8 seconds. If the `BmcRShimEnabled` value does not change, disable BMC RShim by setting the value to `false` then repeating steps 1-3.

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.

Note

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.



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.

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

Note

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

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):

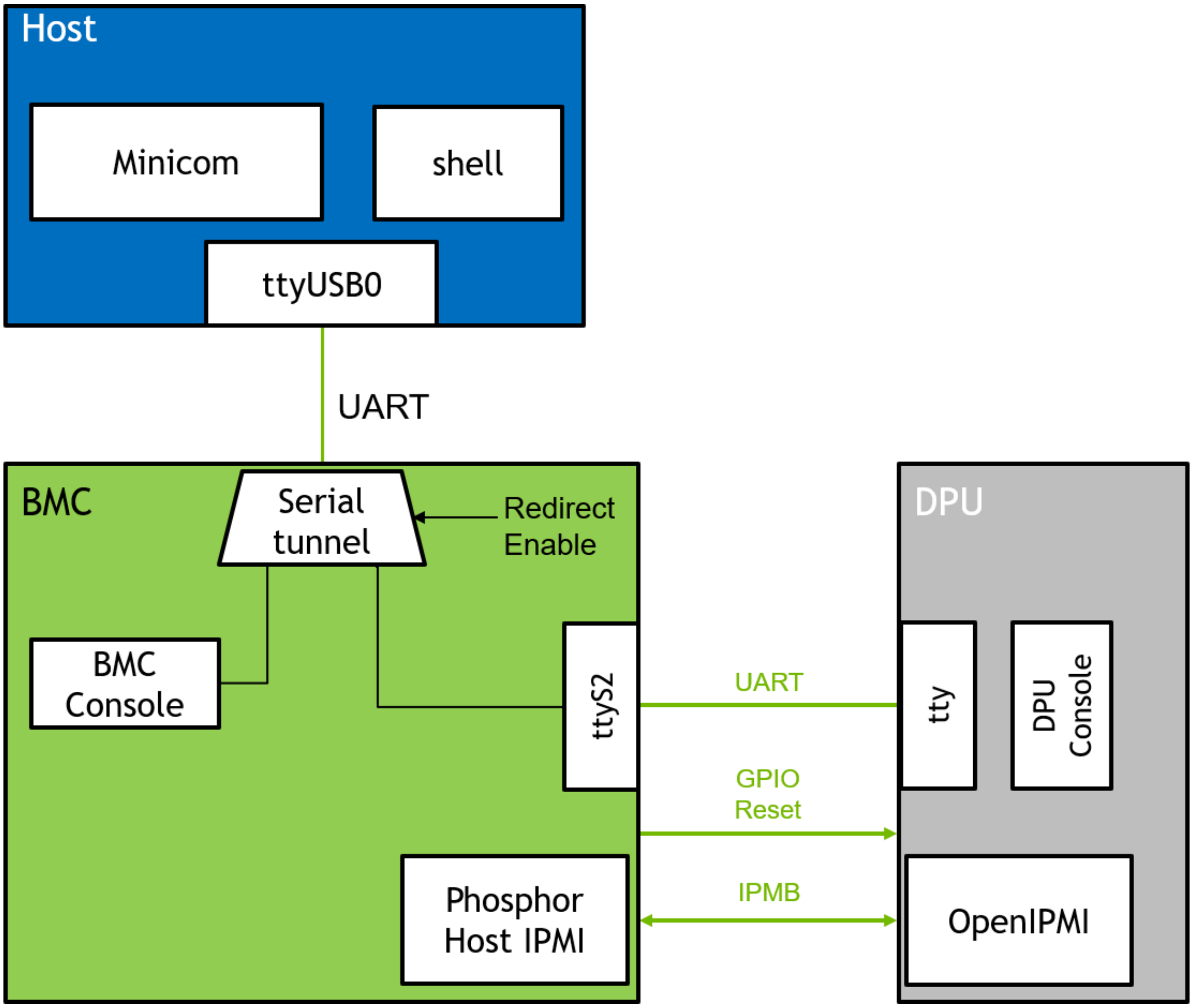
```
ssh -p 2200 root@10.10.10.10
root@10.10.10.10's password:

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)
```

Note

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`).

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.

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
```

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
```

Fetching Serial Redirect Mode Settings

```
ipmitool raw 0x32 0x6E
```

Starting Tunneling on BMC Through UART

Run the following command on the host where BMC is connected**:**

```
/usr/bin/nvidia-field-mode-modifier starttunnel
```

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.

Info

If on an SSH connection, 'escape' the `~` character by entering it twice : `~~.`

Vendor Field Mode

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

Enabling VFM automatically performs the following on BMC:

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

The `fieldmode` user can perform the following operations over UART:

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

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

Disabling VFM automatically performs the following on BMC:

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

Updating BMC Firmware with Vendor Field Mode

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

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

   Expect the following sequence of chars when the tunnel is up and running: 169 150 230.

   Expect the following sequence of chars when the tunnel is not running: 165 200.

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

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

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

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

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

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

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

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

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

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

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

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

8. Check the new BMC firmware version.

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

Updating CEC Firmware with Vendor Field Mode

Note

Relevant only for BlueField-2.

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

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

   Expect the following sequence of characters when the tunnel is up and running: 169 150 230.

   Expect the following sequence of characters when the tunnel is not running: 165 200.

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

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

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

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

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

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

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

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

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

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

Updating BMC and Glacier Firmware with Vendor Field Mode

> **Note**
>
> Relevant only for BlueField-3.

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

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

Expect the following sequence of characters when the tunnel is up and running: 169 150 230.

Expect the following sequence of characters when the tunnel is not running: 165 200.

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

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

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

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

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

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

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

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

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

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

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

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

8. Check the new BMC firmware version.

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

Supported Vendor Field Mode Commands

Operation Description	**Command**
Enable VFM	Run from Arm/BlueField OS and reboot NIC-BMC:

```
ipmitool raw 0x32 0x67 0x01
```

Operation Description	**Command**
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
```

Operation Description	**Command**
Get the status of the tunnel through UART	Run the following command on the host where the BMC is connected:

Operation Description	Command
	`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:

Operation Description	Command
	`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

To check the CEC firmware update status on BMC
Note
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`

Operation Description	Command
	`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
Note
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.

Reset Control

Note

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.

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",
  …
}
```

Hard Reset of BlueField Arm Cores and NIC Subsystem

Note

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/ComputerSyste
-d '{"ResetType" : "PowerCycle"}'
```

Info

Refer to the "RedFish Post (Action)" section in Redfish Specification DSP0266 for an example of successful action response.

Force Hard Reset of BlueField Arm Cores and NIC Subsystem

Info

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.Force
```

Info

Refer to the "RedFish Post (Action)" section in Redfish Specification DSP0266 for an example of successful action response.

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/ComputerSyst
-d '{"ResetType" : "ForceRestart"}'
```

Info

Refer to the "Redfish Post (Action)" section in Redfish Specification DSP0266 for an example of successful action response.

Soft Shutdown of BlueField Arm OS

Note

This command is relevant only for BlueField-3 devices.

Note

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/ComputerSyste
-d '{"ResetType": "GracefulShutdown"}'
```

Info

Refer to the "RedFish Post (Action)" section in Redfish Specification DSP0266 for an example of successful action response.

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"
  },
…
```

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`

Description	Command
	`power reset`
Soft Shutdown of BlueField Arm OS **Note** 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".

Note

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.

Note

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".

Note

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 **Info** 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

Monitoring BlueField OS Shutdown Using IPMI

After a successful shutdown, the BlueField Arm enters a low-power standby state.

Info

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.

ⓘ **Info**

The output of IPMItool chassis power status will show "Chassis power is on".

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.

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

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

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

```
"GracefulRestart"}' https://<BF-BMC-
IP>/redfish/v1/Managers/Bluefield_BMC/Actions/Manager.Reset
{
  "@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The request completed successfully.",
      "MessageArgs": [],
      "MessageId": "Base.1.13.0.Success",
      "MessageSeverity": "OK",
      "Resolution": "None"
    }
  ]
```

Step 2 – Wipe BlueField eMMC and SSD Storage

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

- eMMC:

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

- NVMe:

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

Step 3 – Reset UEFI to Factory Default

Use the Redfish BIOS Settings PATCH command:

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

Step 4 – Reset NIC TLVs to Factory Default

Run the following from the Arm console:

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

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.

Redfish Commands

Note

For detailed information about the DPU attestation process, measurement descriptions, and reference values, refer to the *DPU Attestation* documentation.

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 (Endpoint Root of Trust), which contains measurements related to the DPU BMC

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.

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.SPDMGetSignedMeasureme
```

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.

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"
}
```

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"
}
```

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"
}
```

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"
}
```

Notice

Trademarks