



# System Management

# Table of contents

<b>Common Configurations</b>	4
BlueField Modes of Operation Configuration	4
BIOS Secure Boot Configuration	5
BIOS Configuration	14
<b>Update and Recovery</b>	28
System Inventory	28
Deploying BlueField Software Using BFB from BMC	32
Boot Configuration	61
Modular Update	72
<b>Monitoring</b>	73
System FRU	73
System Logs	75
Retrieving Data from BlueField Via IPMB	136
BMC Sensor Data	140
BlueField Arm State	149
Rsyslog	151
<b>Network</b>	159
BlueField Host Network Interface	159
Factory Reset	162
OOB Network 3-Port Switch Control	165
<b>DPU Information</b>	171
<b>DPU Chassis</b>	172

Reset Control	178
BMC and BlueField Logs	186
Power Capping	197
Serial Over LAN	204
RShim Over USB	210
Vendor Field Mode	212
Serial Redirect Mode	222
Bare-metal Reprovisioning	225

This section contains the following pages:

- [Common Configurations](#)
- [Update and Recovery](#)
- [Monitoring](#)
- [Network](#)
- [DPU Information](#)
- [DPU Chassis](#)
- [Reset Control](#)
- [BMC and BlueField Logs](#)
- [Power Capping](#)
- [Serial Over LAN](#)
- [RShim Over USB](#)
- [Vendor Field Mode](#)
- [Serial Redirect Mode](#)
- [Bare-metal Reprovisioning](#)

---

# 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

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

```
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Oem/Nvidia/Actions/Mo
```

## Setting NIC Mode

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

# 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/Bluefie:
d '{"SecureBootEnable":true}'

```

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

```

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

```

```
application/json' -d '{"SecureBootEnable": false}'
```

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

- From the BlueField BMC using Redfish:

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

- From RShim:

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

- From the BlueField Arm OS:

```
reboot
```

## 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/Secure
```

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/
    },
    ..
  ]
}
```



```
}
```

- 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/SecureBoot/
-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/SecureBoot/
```

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/SecureBoot
-d '{"ResetKeyType": "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 resetting all `db` certificates using Redfish commands:

1. To reset all `db` keys:



```

"@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  \"ResetKeyType\":\n  \n\"DeleteAllKeys\"\n}\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.

### Getting BIOS Attributes List

```
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`.

```
{
  "@odata.type": "#AttributeRegistry.v1_3_6.AttributeRegistry",
  "Description": "This registry defines a representation of BIOS
attribute instances",
  "Id": "BiosAttributeRegistry",
  "Language": "en",
  "Name": "BF2 BIOS Attribute Registry",
  "OwningEntity": "NVIDIA BlueField",
  "RegistryEntries": {
    "Attributes": [
      {
        "AttributeName": "UefiPassword",
        "CurrentValue": "",
        "DisplayName": "New UEFI Password",
        "DisplayOrder": 16,
        "HelpText": "Set the UEFI password.",
        "Hidden": false,
        "Immutable": false,
        "MaxLength": 50,
        "MenuPath": "./SystemConfiguration/UefiPassword",
        "MinLength": 0,
        "ReadOnly": false,
        "ResetRequired": false,
        "Type": "String",
        "WriteOnly": false
      }
    ]
  }
}
```

```

    }
  ],
  "Dependencies": [],
  "Menus": [
    {
      "DisplayName": "Set the UEFI Password.",
      "DisplayOrder": 18,
      "Hidden": false,
      "MenuName": "UefiPassword",
      "MenuPath": "./SystemConfiguration/UefiPassword",
      "ReadOnly": false
    }
  ]
},
"RegistryVersion": "1.0.0",
"SupportedSystems": [
  {
    "FirmwareVersion": "BlueField:4.2.0-33-gbe969d4",
    "ProductName": "NVIDIA BF2",
    "SystemId": "BF2-DPU"
  }
]
}

```

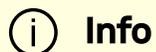
## Getting Current BIOS Attributes Value

```

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

The following command example requests changing the `UefiPassword` attribute to `NewPassword123`:

```
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:

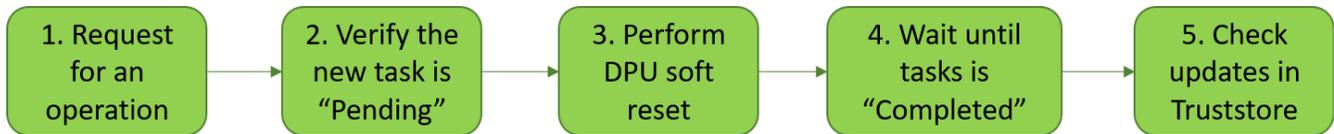
```
curl -k -u root:<password> -X GET
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Oem/Nvidia/Truststore/Certificates/<cert_num>
```

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": "Kg639lcpJtYMRzvh.nvidia",
    "Country": "US",
    "Organization": "NVIDIA",
    "OrganizationalUnit": "NBU",
    "State": "California"
  },
  "KeyUsage": [
    "CRLSigning"
  ],
  "Name": "TruststoreBios Certificate",
  "Subject": {
    "City": "Santa Clara",
    "CommonName": "Kg639lcpJtYMRzvh.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:

- 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/0em/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.ResetKeysType/DeleteAllKeys'
```

2. Verify the new task is **Pending**:

The responses of these requests indicate the creation of a new task for the UEFI:

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

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

```
curl -k -u root:<password> -H "Content-Type: application/json" -X POST https://<bmc_ip>/redfish/v1/Systems/Bluefield/Actions/ComputerSystem.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 BlueField Software Using BFB from BMC](#)
- [Boot Configuration](#)
- [Modular Update](#)

## 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_BOARD"
    },
    {
      "@odata.id":
"/redfish/v1/UpdateService/FirmwareInventory/DPU_BSP"
    },
    {
      "@odata.id":
"/redfish/v1/UpdateService/FirmwareInventory/DPU_NIC"
    },
    {
      "@odata.id":
"/redfish/v1/UpdateService/FirmwareInventory/DPU_NODE"
    },
  ],
}
```

```

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

```

- 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/DPU
```

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 BlueField Software Using BFB from BMC

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

## Changing Default Credentials Using bf.cfg

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

### Note

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

Alternatively, Ubuntu users can provide a unique password that will be applied at the end of the BFB installation. This password 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$3B0RIrfX$T1Hry93NFUJzg3Nya00rE1
```

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

```
# vim bf.cfg
```

```
ubuntu_PASSWORD=' $1$3B0RIrfX$T1Hry93NFUJzg3Nya00rE1 '
```

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

## Installing BFB

The BFB installation procedure consists of the following main stages:

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

### Note

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

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

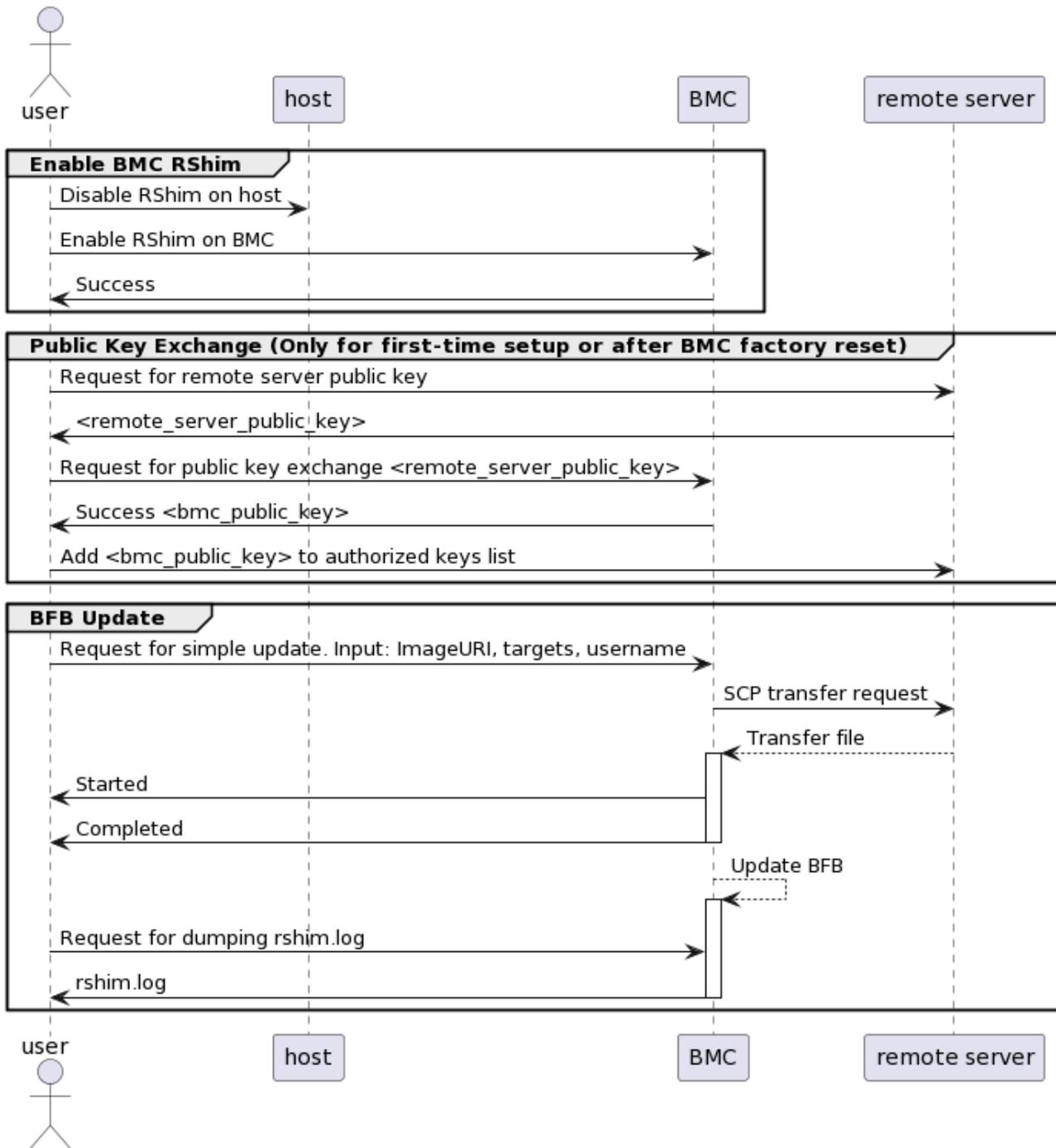
# Transferring BFB File

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

## Redfish Interface

### Installing BFB File Using SCP Protocol

## BMC Image Update Flow Using UpdateService POST Command



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

1. Prepare secure file transfer of BFB:

**Note**

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

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

**i Info**

OpenSSH is required for this step.

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

Where:

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

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

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

Where:

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

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

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

```
}  
]  
}
```

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

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

### **(i) Note**

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

### **(i) Info**

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

Where:

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

### Info

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

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

Response/error messages:

- If RShim is disabled:

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

```

    }
  ],
  "code": "Base.1.15.0.ResourceNotFound",
  "message": "The requested resource of type
Target named '/dev/rshim0/boot' was not found."
}

```

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

```

{
  "Username@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The create operation failed
because the required property Username was missing
from the request.",
      "MessageArgs": [
        "Username"
      ],
      "MessageId":
"Base.1.15.0.CreateFailedMissingReqProperties",
      "MessageSeverity": "Critical",
      "Resolution": "Correct the body to include the
required property with a valid value and resubmit
the request if the operation failed."
    }
  ]
}

```

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

```

{
  "@odata.id":

```

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

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

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

### **(i) Note**

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

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

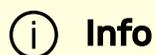
Where:

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

Examples:

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

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



**Info**

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

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

Where:

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

Then repeat steps 1 and 2.

- - -
- If the BMC identity is not confirmed:

```
{
  "@odata.type":
"#MessageRegistry.v1_4_1.MessageRegistry",
  "Message": "Transfer of image
'<file_name>' to '/dev/rshim0/boot' failed.",
  "MessageArgs": [
    "<file_name>",
    "/dev/rshim0/boot"
  ],
  "MessageId": "Update.1.0.TransferFailed",
```

```

        "Resolution": "Unauthorized Client: Please
use the PublicKeyExchange action to receive the
system's public key and add it as an authorized
key on the remote server",
        "Severity": "Critical"
    }
...
    "PercentComplete": 0,
    "StartTime": "<start_time>",
    "TaskMonitor":
"/redfish/v1/TaskService/Tasks/<task_id>/Monitor",
    "TaskState": "Exception",
    "TaskStatus": "Critical"

```

### Info

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

- 
- 
- 
- If SCP fails:

```

{
    "@odata.type":
"#MessageRegistry.v1_4_1.MessageRegistry",
    "Message": "Transfer of image
'<file_name>' to '/dev/rshim0/boot' failed.",
    "MessageArgs": [
        "<file_name>",

```

```

        "/dev/rshim0/boot"
    ],
    "MessageId": "Update.1.0.TransferFailed",
    "Resolution": "Failed to launch SCP",
    "Severity": "Critical"
}
...
"PercentComplete": 0,
"StartTime": "<start_time>",
"TaskMonitor":
"/redfish/v1/TaskService/Tasks/<task_id>/Monitor",
"TaskState": "Exception",
"TaskStatus": "Critical"

```

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

```

{
    "@odata.type":
"#MessageRegistry.v1_4_1.MessageRegistry",
    "Message": " <file_name>' is being
transferred to '/dev/rshim0/boot'.",
    "MessageArgs": [
        " <file_name>",
        "/dev/rshim0/boot"
    ],
    "MessageId":
"Update.1.0.TransferringToComponent",
    "Resolution": "Transfer is still in
progress (5 minutes elapsed): Please wait",

```

```

        "Severity": "OK"
    }
...
"PercentComplete": 0,
  "StartTime": "<start_time>",
  "TaskMonitor":
"/redfish/v1/TaskService/Tasks/<task_id>/Monit
  "TaskState": "Running",
  "TaskStatus": "OK"

```

- Upon successful completion of SCP BFB transfer:

```

{
    "@odata.type":
"#MessageRegistry.v1_4_1.MessageRegistry",
    "Message": "Device 'DPU' successfully
updated with image '<file_name>'.",
    "MessageArgs": [
        "DPU",
        "<file_name>"
    ],
    "MessageId":
"Update.1.0.UpdateSuccessful",
    "Resolution": "None",
    "Severity": "OK"
},
...
"PercentComplete": 100,
  "StartTime": "<start_time>",
  "TaskMonitor":
"/redfish/v1/TaskService/Tasks/<task_id>/Monit
  "TaskState": "Completed",
  "TaskStatus": "OK"

```

## Installing BFB File with HTTP Protocol

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

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

### Note

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

### Info

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

Where:

-

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

### Info

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

- `bmc_ip` – BMC IP address

Response/error messages:

- If RShim is disabled:

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

```
    "code": "Base.1.15.0.ResourceNotFound",
    "message": "The requested resource of type
Target named '/dev/rshim0/boot' was not found."
}
```

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

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

- If `TransferProtocol` or any other required field are wrong:

```
{
  "@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The parameter TransferProtocol
for the action UpdateService.SimpleUpdate is not
supported on the target resource.",

```

```

        "MessageArgs": [
            "TransferProtocol",
            "UpdateService.SimpleUpdate"
        ],
        "MessageId":
"Base.1.16.0.ActionParameterNotSupported",
        "MessageSeverity": "Warning",
        "Resolution": "Remove the parameter supplied
and resubmit the request if the operation failed."
    }
]
}

```

- If `Targets` or any other required field are missing:

```

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

```

```
}
```

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

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

## Installing BFB File with HTTPS Protocol

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

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

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

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

```
["redfish/v1/UpdateService/FirmwareInventory/DPU_OS" ]}'  
https://<bmc_ip>/redfish/v1/UpdateService/Actions/UpdateService
```

### Note

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

### Info

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

Where:

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

### Info

For example, if `<https_server>` is [urm.nvidia.com](https://urm.nvidia.com) and `<exported_path_of_bfb>` is `artifactory/sw-mlnx-bluefield-generic/Ubuntu22.04/new.bfb` then

```
"ImageURI": "10.126.206.42/artifactory/sw-mlnx-bluefield-generic/Ubuntu22.04/new.bfb"
```

- `bmc_ip` – BMC IP address

Response / error messages:

- - - If RShim is disabled:

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

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

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

- If `TransferProtocol` or any other required field are wrong:

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

```

        "MessageId" :
"Base.1.16.0.ActionParameterNotSupported",
        "MessageSeverity": "Warning",
        "Resolution": "Remove the parameter supplied
and resubmit the request if the operation failed."
    }
]
}

```

- - - If `Targets` or any other required field are missing:

```

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

```

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

```
{
  "@odata.type":
"#MessageRegistry.v1_4_1.MessageRegistry",
  "Message": "Transfer of image 'new.bfb' to
'/dev/rshim0/boot' failed.",
  "MessageArgs": [
    "new.bfb",
    "/dev/rshim0/boot"
  ],
  "MessageId": "Update.1.0.TransferFailed",
  "Resolution": "Bad Certificate: Please check
the remote server certification, correct and replace
the current installed one",
  "Severity": "Critical"
},
```

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

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

## Tracking Image Transfer Status and Progress for HTTP/HTTPS Protocols

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

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

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

Example:

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

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

## Direct SCP

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

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

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

## Tracking Installation Progress and Status

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

1. Initiate request for dump download:

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

Where:

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

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

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

Where:

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

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

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

Where:

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

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

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

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

### **Note**

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

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

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

Example response:

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

```
        "@odata.id": "/redfish/v1/Systems/Bluefield/Bios"
    }
  ],
  "SoftwareId": "",
  "Status": {
    "Conditions": [],
    "Health": "OK",
    "HealthRollup": "OK",
    "State": "Enabled"
  },
  "Updateable": true,
  "Version": "DOCA_2.2.0_BSP_4.2.1_Ubuntu_22.04-8.23-07"
```

## 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 only supports the option to change the boot device selector flag with the following supported options: PXE boot or the default boot device as selected in the boot menu on BlueField. While 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/<boot
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.

## **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"] }}}
```

## Example of Changing Boot Configuration

To set boot configuration, it is necessary to post to settings. For example:

```
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"
}
}'
```

- `BootSourceOverrideEnabled` – should be set according to the values under `redfish/v1/Systems/Bluefield` >

Boot/BootSourceOverrideEnabled@Redfish.AllowableValues

- `BootSourceOverrideMode` – should be set according to the values under `redfish/v1/Systems/Bluefield` >  
`Boot/BootSourceOverrideMode@Redfish.AllowableValues`
- `BootSourceOverrideTarget` – should be set according to the values under `redfish/v1/Systems/Bluefield` >  
`Boot/BootSourceOverrideTarget@Redfish.AllowableValues`

### **(i) Note**

When `UefiTarget` is selected, make sure that:

- `BootSourceOverrideMode` is set to `UEFI`
- `UefiTargetBootSourceOverride` is set to one of the UEFI supported `BootOptions` (e.g., `Boot0007`)

- `UefiTargetBootSourceOverride` – this option would be available in the RF JSON if `BootSourceOverrideTarget` is set to `UefiTarget`
- `BootNext` – this option would be in the RF JSON if `BootSourceOverrideTarget` is set to `UefiBootnext`
- `AutomaticRetryConfig` – only `Disabled` is supported

## **Boot Config Using IPMI**

The `ipmitool` only provides the ability to manage the override boot option and configure the system to boot from a PXE server.

- Get current setting:

```
ipmitool chassis bootparam get 5
```

- Force PXE boot:

```
ipmitool chassis bootdev pxe options=efiboot
```

- Default boot device:

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

### Info

The BlueField boot override setting from BMC is persistent until it is set to `none` or the BFB image is updated again.

## Modular Update

Please refer to the "Upgrading BlueField Using Standard Linux Tools" section under the [\*NVIDIA DOCA Installation Guide for Linux\*](#).

---

# Monitoring

This section contains the following pages:

- [System FRU](#)
- [System Logs](#)
- [Retrieving Data from BlueField Via IPMB](#)
- [BMC Sensor Data](#)
- [BlueField Arm State](#)
- [Rsyslog](#)

## 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": "Nvidia",
  "Model": "BlueField-3 DPU",
  "PartNumber": "900-9D3B4-00EN-EAB  ",
}
```

```
"SerialNumber": "MT2245X00175",  
}
```

## FRU Reading IPMI Commands

To retrieve FRU info, run:

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

The System FRU ID 0: Contain information for NVIDIA® BlueField® System:

```
FRU Device Description : Builtin FRU Device (ID 0)  
Chassis Type           : Main Server Chassis  
Chassis Part Number    : 900-9D3B6-00CV-AAA  
Chassis Serial         : N/A  
Chassis Extra          : N/A  
Chassis Extra          : 1.0.0  
Chassis Extra          : https://www.mellanox.com  
Chassis Area Checksum  : OK  
Board Mfg Date         : Wed Nov  8 01:41:00 2023 UTC  
Board Mfg              : https://www.mellanox.com  
Board Product          : BlueField SoC  
Board Serial           : N/A  
Board Part Number      : 900-9D3B6-00CV-AAA  
Board Extra            : 1.0.0  
Board Area Checksum    : OK  
Product Manufacturer   : https://www.mellanox.com  
Product Name          : BlueField SoC  
Product Part Number    : 900-9D3B6-00CV-AAA  
Product Serial         : MT2345300006  
Product Asset Tag      : N/A  
Product Extra         : 1.0.0
```

Product Area Checksum : OK

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/attach",  
      "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/Nvidia  
-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
```

## 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
```

## 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    : Inc- unc+
Deassertions Enabled  : Inc+ 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
```

## 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 #0x1e | 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
```

### Data Port Module Thermal "Going High" Notification

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

SEL messages:

```
Temperature #0x1d | 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
```

## Data Port Module Thermal "Going Low" Notification

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

SEL messages:

```
Temperature #0x1d | 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
```

## 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/attach",
  "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/attach",
  "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/attach",
  "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/attach",
  "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/attach",
  "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/attach",
  "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"
},
```

## 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/attach
    "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/attach
  "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/attach",
  "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/attach",
  "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/att
  "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/att
  "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/attach",
  "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/attach",  
    "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/attach",  
  "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/attach",
  "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/attach",
  "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/att
  "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/att
  "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/attach",
  "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/attach",  
    "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/attach",  
  "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/att
  "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/att
  "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/attach",
  "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/attach",
  "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/attach",
  "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/attac
    "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/attach",
  "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/attach",
  "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/attac
  "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/att
    "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/att
  "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/attach",
  "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/attach",
  "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/attach",
  "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/attach",
  "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/attach",  
  "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/att
  "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/attac
    "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/attac
    "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/attach",
  "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/attach",
  "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/attach",
  "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/attach",
  "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/attach",
  "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/attachments/CPER",
  "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/attachments/6",
  "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"
}

```

## Retrieving Data from BlueField Via IPMB

The BMC can retrieve information on NVIDIA® BlueField®'s sensors and FRUs via IPMI over IPMB protocol. IPMITool commands can be issued from the BMC using the following format:

```
ipmitool -I ipmb <ipmitool_arguments>
```

## List of IPMI Supported Sensors

Sensor	Sensor ID	Description
<code>bluefield_temp</code>	0	Support NIC monitoring of BlueField's temperature
<code>p0_temp</code>	5	Port 0 temperature
<code>p1_temp</code>	6	Port 1 temperature
<code>p0_link</code>	7	Port0 link status <ul style="list-style-type: none"> <li>• 0x100 – connection OK</li> <li>• 0x200 – connection error</li> </ul>
<code>p1_link</code>	8	Port1 link status <ul style="list-style-type: none"> <li>• 0x100 – connection OK</li> <li>• 0x200 – connection error</li> </ul>

## List of IPMI Supported FRUs

FRU	ID	Description
<code>update_timer</code>	0	<p><code>set_emu_param.service</code> is responsible for collecting data on sensors and FRUs every 3 seconds. This regular update is required for sensors but not for FRUs whose content is less susceptible to change. <code>update_timer</code> is used to sample the FRUs every hour instead. Users may need this timer if they are issuing several raw IPMITool FRU read commands. This helps assess how many times users must retrieve large FRU data before the next FRU update.</p> <p><code>update_timer</code> is a hexadecimal number.</p>

FRU	ID	Description
fw_info	1	ConnectX firmware information, Arm firmware version, and MLNX_OFED version The <code>fw_info</code> is in ASCII format
nic_pci_dev_info	2	NIC vendor ID, device ID, subsystem vendor ID, and subsystem device ID The <code>nic_pci_dev_info</code> is in ASCII format
cpuinfo	3	CPU information reported in <code>lscpu</code> and <code>/proc/cpuinfo</code> The <code>cpuinfo</code> is in ASCII format
emmc_info	8	eMMC size, list of its partitions, and partitions usage (in ASCII format). eMMC CID, CSD, and extended CSD registers (in binary format). The ASCII data is separated from the binary data with <code>StartBinary</code> marker.
qsfp0_eepr om	9	FRU for QSFP 0 EEPROM page 0 content (256 bytes in binary format)
qsfp1_eepr om	10	FRU for QSFP 1 EEPROM page 0 content (256 bytes in binary format)  <div style="background-color: #ffffcc; padding: 10px;"> <p> <b>Info</b> Applicable for dual-port devices only.</p> </div>

FRU	ID	Description
ip _a dd re ss es	1 1	<p>This FRU is empty at start time. It can be used to write the BMC port 0 and port 1 IP addresses to the BlueField. They follow these formats:</p> <pre>BMC : XXX.XXX.XXX.XXX P0 : XXX.XXX.XXX.XXX P1 : XXX.XXX.XXX.XXX</pre> <p>The size of the written file should be 61 bytes exactly.</p>
et h0	1 3	Network interface 0 information. Updated once every minute.
et h1	1 4	<p>Network interface 1 information. Updated once every minute.</p> <p><b>i Info</b> Applicable for dual-port devices only.</p>
bf _u id	1 5	BlueField device UUID
et h_ hw _c ou nt er s	1 6	List of ConnectX interface hardware counters
oo b0	1 7	Out-Of-Band port network interface 0 information. Updated once every minute.
bf _f ru	1 8	FRU information of the BlueField

FRU	ID	Description
product_name	19	The BlueField product name
dmidecode_info	20	Dmidecode information (e.g., part number, product name)

## Supported IPMI Commands

All the following commands are prepended with `ipmitool` on the command line.

Commands	IPMITool Command	Relevant IPMI 2.0 Rev1.1 Spec Section
Get Device ID	<code>mc info</code>	20.1
Broadcast "Get Device ID"	Part of <code>"mc info"</code>	20.9
Get BMC Global Enables	<code>mc getenables</code>	22.2
Get Device SDR Info	<code>sdr info</code>	35.2
Get Device SDR	<code>"sdr get", "sdr list" or "sdr elist"</code>	35.3
Get Sensor Hysteresis	<code>sdr get &lt;sensor_id&gt;</code>	35.7
Set Sensor Threshold	<code>sensor thresh &lt;sensor-id&gt; &lt;threshold&gt; &lt;setting&gt;</code>	35.8

Commands	IPMItool Command	Relevant IPMI 2.0 Rev1.1 Spec Section
Get Sensor Threshold	<code>sdr get &lt;sensor_id&gt;</code>	35.9
Get Sensor Event Enable	<code>sdr get &lt;sensor_id&gt;</code>	35.11
Get Sensor Reading	<code>sensor reading &lt;sensor_id&gt;</code>	35.14
Get Sensor Type	<code>sdr type &lt;type&gt;</code>	35.16
Read FRU Data	<code>fru read &lt;fru_number&gt;</code> <code>&lt;file_to_write_to&gt;</code> - provides FRU data	34.2
Get SDR Repository Info	<code>sdr info</code>	33.9
Get SEL Info	<code>"sel"</code> or <code>"sel info"</code>	40.2
Get SEL Allocation Info	<code>"sel"</code> or <code>"sel info"</code>	40.3
Get SEL Entry	<code>"sel list"</code> or <code>"sel elist"</code>	40.5
Delete SEL Entry	<code>sel delete &lt;id&gt;</code>	40.8
Clear SEL	<code>sel clear</code>	40.9

## 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
<code>p0_link</code>	Discrete	IPMB	Uplink port 0 link status

Sensor Name	Sensor Type	Source	Description
			<ul style="list-style-type: none"> <li>0x100 – connection OK</li> <li>0x200 – connection error</li> </ul>
p1_link	Discrete	IPMB	Uplink port 1 link status <ul style="list-style-type: none"> <li>0x100 – connection OK</li> <li>0x200 – connection error</li> </ul>
bluefield_temp	Temperature	IPMB	NVIDIA® BlueField® temperature
p0_temp	Temperature	IPMB	Uplink port 0 SFP temperature
p1_temp	Temperature	IPMB	Uplink port 1 SFP temperature
ddr_temp	Temperature	IPMB	DDR temperature
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 (power from gold fingers in case of Sub75 when ATX power is off)

Sensor Name	Sensor Type	Source	Description
12V_PCIE	Voltage	BMC ADC	Input power rail from gold fingers
DVDD	Voltage	BMC ADC	
HVDD	Voltage	BMC ADC	
VDD	Voltage	BMC ADC	
VDDQ	Voltage	BMC ADC	
VDD_CPU_L	Voltage	BMC ADC	
VDD_CPU_R	Voltage	BMC ADC	

## 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/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"
  }
],
"Members@odata.count": 20,
"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.

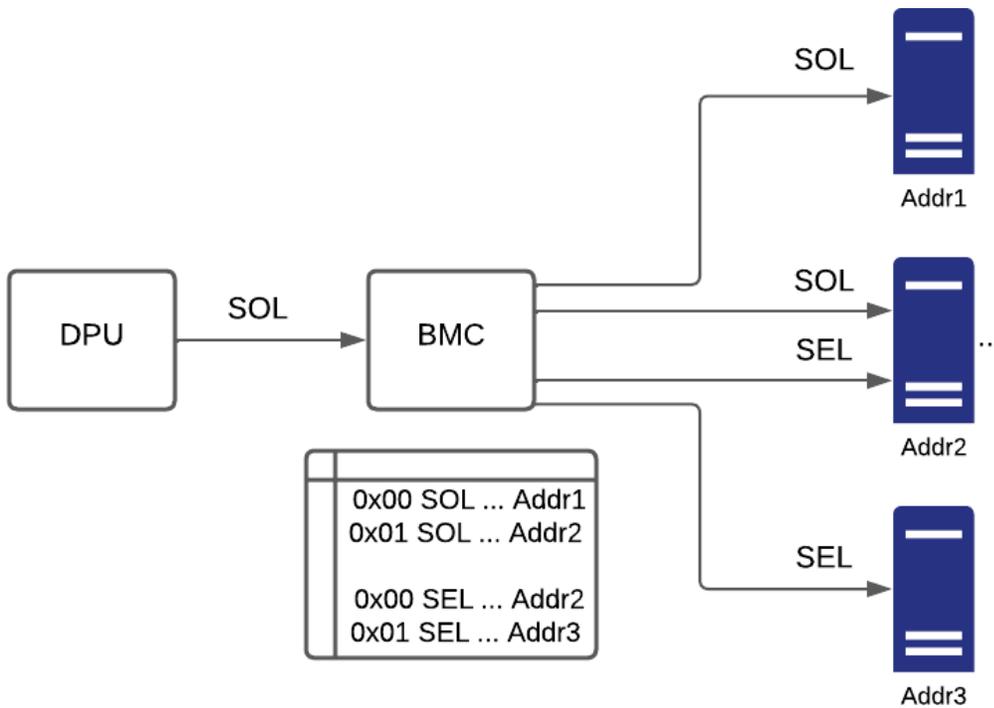
## Rsyslog Servers Configurations

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:

net func	cmd	data	Description
0x32	0xD3	<Index> <LogType>	<p>Get rsyslog status – Displays information of the configured rsyslog server</p> <p>The request contains the index and the log type of the rsyslog server configuration, and it is 2 bytes long.</p> <p>The response contains the following information:</p> <pre> &lt;Index&gt; &lt;LogType&gt; &lt;Status&gt; &lt;TransportProtocol&gt; &lt;NetworkProtocol&gt; &lt;ServerAddress&gt; &lt;Port&gt; </pre> <ul style="list-style-type: none"> <li>• Byte 1 – Completion code: <ul style="list-style-type: none"> <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 style="list-style-type: none"> <li>◦ Index of server (0x00-0x09)</li> </ul> </li> <li>• Byte 3 – LogType <ul style="list-style-type: none"> <li>◦ 0x01 – SEL</li> <li>◦ 0x03 – SOL</li> </ul> </li> <li>• Byte 4 – Status <ul style="list-style-type: none"> <li>◦ 0x00 – Disabled</li> <li>◦ 0x01 – Enabled</li> </ul> </li> <li>• Byte 5 – Transport protocol <ul style="list-style-type: none"> <li>◦ 0x00 – UDP</li> <li>◦ 0x01 – TCP</li> </ul> </li> <li>• Byte 6 – Network protocol <ul style="list-style-type: none"> <li>◦ 0x00 – IPv4</li> <li>◦ 0x01 – IPv6</li> </ul> </li> <li>• Byte 7-n – Rsyslog server address <ul style="list-style-type: none"> <li>◦ Rsyslog addr (4/16 Bytes)</li> </ul> </li> <li>• Byte n+1-n+2 – Port <ul style="list-style-type: none"> <li>◦ Rsyslog port. LSB first.</li> </ul> </li> </ul>

net func	cmd	data	Description
			<p>The response is 12 bytes long for IPv4 and 24 bytes long for IPv6.</p>
<p>0x32</p>	<p>0xD4</p>	<p>&lt;Index&gt; &lt;LogType&gt; &lt;Status&gt; &lt;TransportProtocol&gt; &lt;NetworkProtocol&gt; &lt;ServerAddress&gt; &lt;Port&gt;</p>	<p>Set rsyslog status –</p> <ul style="list-style-type: none"> <li>Configures a new rsyslog server configuration if the configuration &lt;Index&gt; &lt;LogType&gt; does not exist.</li> <li>Modifies an existing rsyslog server configuration if the configuration &lt;Index&gt; &lt;LogType&gt; does exist.</li> </ul> <p>The command contains the following information:</p> <ul style="list-style-type: none"> <li>Byte 1 – Index <ul style="list-style-type: none"> <li>Index of server (0x00-0x09)</li> </ul> </li> <li>Byte 2 – LogType <ul style="list-style-type: none"> <li>0x01 – SEL</li> <li>0x03 – SOL</li> </ul> </li> <li>Byte 3 – Status <ul style="list-style-type: none"> <li>0x00 – Disabled</li> <li>0x01 – Enabled</li> </ul> </li> <li>Byte 4 - Transport protocol <ul style="list-style-type: none"> <li>0x00 – UDP</li> <li>0x01 – TCP</li> </ul> </li> <li>Byte 5 - Network protocol <ul style="list-style-type: none"> <li>0x00 – IPv4</li> <li>0x01 – IPv6</li> </ul> </li> <li>Byte 6-n – Rsyslog server address <ul style="list-style-type: none"> <li>Rsyslog addr (4/16 Bytes)</li> </ul> </li> <li>Byte n+1-n+2 – Port <ul style="list-style-type: none"> <li>Rsyslog port. LSB first.</li> </ul> </li> </ul> <p>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.</p>

# Usage Examples

## Setting Rsyslog Status of Two Configurations

The following commands create or modify two different rsyslog configurations with Index 0x00 and LogTypes SEL/SOL :

netfunc: 0x32, cmd: 0xD4, Indx: 0x00, LogType: 0x01(SEL) / 0x03(SOL), status: 0x01 (Enabled), TP: 0x01 (TCP), NP: 0x00 (IPv4) Address: 0x0A 0xED 0x33 0xF4 (10.237.51.244) Port: 0xFA 0x13 (5114)

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

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

The following command disables the rsyslog configurations with Index 0x00 and LogTypes SOL:

netfunc: 0x32, cmd: 0xD4, Indx: 0x00, LogType: 0x03 (SOL), status: 0x00 (Disabled), TP: 0x01 (TCP), NP: 0x00 (IPv4) Address: 0x0A 0xED 0x33 0xF4 (10.237.51.244) Port: 0xFA 0x13 (5114)

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

Now the rsyslog server receives only SEL messages as the SOL configuration is disabled:

netfunc: 0x32, cmd: 0xD3, Indx: 0x00, LogType: 0x01(SEL) / 0x03(SOL)

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

```
00 03 00 01 00 0a ed 33 f4 fa 13
```

## 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
```

---

# Network

This section contains the following pages:

- [BlueField Host Network Interface](#)
- [Factory Reset](#)
- [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).

### 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/oob0
{
  "@odata.id" :
"/redfish/v1/Systems/Bluefield/EthernetInterfaces/oob0",
  "@odata.type" : "#EthernetInterface.v1_6_0.EthernetInterface",

```

```
"DHCPv4": {
  "DHCPEnabled": true,
  "UseDNSServers": false,
  "UseDomainName": false,
  "UseNTPServers": false
},
"DHCPv6": {
  "OperatingMode": "Stateful",
  "UseDNSServers": false,
  "UseDomainName": false,
  "UseNTPServers": false
},
"Description": "Host Network Interface for port oob0",
"IPv4Addresses": [
  {
    "Address": "10.345.41.97",
    "AddressOrigin": "Static",
    "Gateway": "0.0.0.0",
    "SubnetMask": "255.255.240.0"
  }
],
"IPv4StaticAddresses": [
  {
    "Address": "10.345.41.97",
    "AddressOrigin": "Static",
    "Gateway": "0.0.0.0",
    "SubnetMask": "255.255.240.0"
  }
],
"IPv6AddressPolicyTable": [],
"IPv6Addresses": [
  {
    "Address": "fe80::a278:c2ff:fe0e:87a4",
    "AddressOrigin": "Static",
    "AddressState": null,
    "PrefixLength": 64
  }
]
```

```

    }
  ],
  "IPv6DefaultGateway": "0:0:0:0:0:0:0:0",
  "IPv6StaticAddresses": [
    {
      "Address": "fe80::a278:c2ff:fe0e:87a4",
      "PrefixLength": 64
    }
  ],
  "Id": "oob0",
  "InterfaceEnabled": true,
  "LinkStatus": "LinkUp",
  "MACAddress": "a0:88:a2:0e:87:a4",
  "MTUSize": 1500,
  "Name": "Host Ethernet Interface",
  "NameServers": [],
  "SpeedMbps": 1000,
  "StaticNameServers": [],
  "Status": {
    "State": "Disabled"
  }
}

```

### **Note**

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

## 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
```

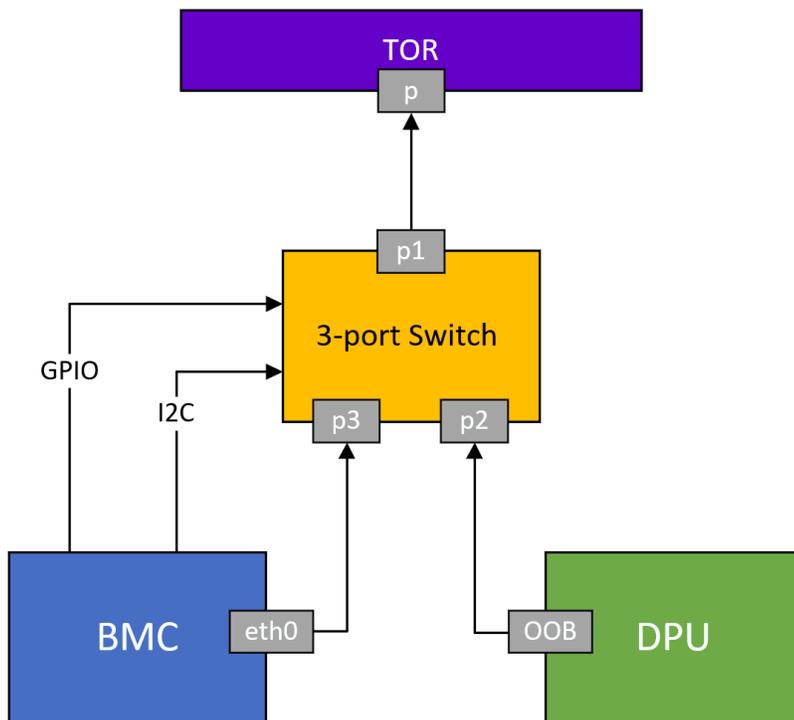
## Step 4 – Reset NIC TLVs to Factory Default

Run the following from the Arm console:

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

# 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	<p>Get 3-port switch ports mode. On success, it returns:</p> <ul style="list-style-type: none"> <li>• 0x00 – all ports are allowed access to RJ45</li> <li>• 0x01 – only BMC is allowed access to RJ45</li> </ul>
0x32	0x98	<ul style="list-style-type: none"> <li>• 0x00 – all ports are allowed access to RJ45</li> <li>• 0x01 – only BMC is allowed access to RJ45</li> </ul>	<p>Set 3-port switch ports mode.</p> <ul style="list-style-type: none"> <li>• 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)</li> <li>• Setting is persistent across power cycle and switch reset command</li> </ul>
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/0em/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.Reset
```

Example output:

```
{
```

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

---

# DPU Information

## 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'
```

---

# 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

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

- Ports

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

```

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

---

# Reset Control

## Note

Rebooting 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

### Info

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

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

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

### 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.ForceReset
```

### 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/ComputerSystem.ResetType  
-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.

```
curl -k -u root:'<password>' -H "Content-Type: application/json"  
-X POST  
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Actions/ComputerSystemReset  
-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 of BlueField, use the following commands:

Description	Command
Hard reset of BlueField (Arm cores and NIC)	<pre>ipmitool chassis power cycle</pre>
Hard reset of BlueField Arm cores	<pre>ipmitool chassis power reset</pre>

Description	Command
<p data-bbox="159 226 686 258">Soft Shutdown of BlueField Arm OS</p> <div data-bbox="186 359 763 485"> <p><b>Note</b> This command is relevant only for BlueField-3.</p> </div>	<pre data-bbox="1024 331 1463 489">ipmitool power soft</pre>

**Note**

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

- PERST signal on BlueField-2
- ALL\_STANDBY signal on BlueField-3

**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
<ul style="list-style-type: none"> <li>• <code>0x32</code> – NetFun</li> <li>• <code>0xA1</code> – command</li> <li>• <code>0x00</code> – Req_data1 (reset option)</li> </ul>	<p>Completion code:</p> <ul style="list-style-type: none"> <li>• <code>0x00</code> – success</li> <li>• <code>&lt;ipmi-error-code&gt;</code> – failure</li> </ul>	<ul style="list-style-type: none"> <li>• <code>0x02</code> – soft reset of BlueField Arm cores</li> </ul> <div style="background-color: #ffffcc; padding: 10px; border: 1px solid #ccc;"> <p><b>i Info</b> This reset command is only available when the BlueField Arm OS is up.</p> </div> <ul style="list-style-type: none"> <li>• <code>0x03</code> – reset on-board 3-port switch</li> </ul>

## Monitoring BlueField OS Shutdown Using IPMI

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

### **i 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".

---

# 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

### **(i) Note**

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

### **(i) 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

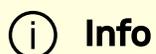
## Downloading BMC Dump

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`
  - 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`

- `iplink.log`
- `ipaddr.log`
- `interrupts.log`
- `freemem.log`
- `channelconfig.log`
- `channelaccess.log`
- `arptable.log`
- `obmc-console.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`
- `procfld.log`
- `dreport.log`
- `disk-usage.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/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/<entry_id> --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/<entry_id>`
- `</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`
- `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/dpu1ogs/`. The log is rotated if it is larger than 1M or older than 24 hours. The oldest console output is overwritten as new data is added.

---

# Power Capping

## 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"
}
}
```

Power consumption information:

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

Output example:

```
{
  "@odata.id": "/redfish/v1/Chassis/Card1/PowerSubsystem",
  "@odata.type": "#PowerSubsystem.v1_1_0.PowerSubsystem",
  "Allocation ": {
    "AllocatedWatts": 200,
    "RequestedWatts": 30,
  },
  "CapacityWatts ": 300,
  "Id": "PowerSubsystem",
  "Name": "Power Subsystem",
  "PowerSupplies": {
    "@odata.id":
"/redfish/v1/Chassis/Card1/PowerSubsystem/PowerSupplies"
  },
  "Status": {
    "Health": "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, then the UEFI sets the system power to minimum capacity.

## 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
```

### **Note**

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

## 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.

## Getting Capacity Allocation

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

The amount of power allocated to the system in Watts.

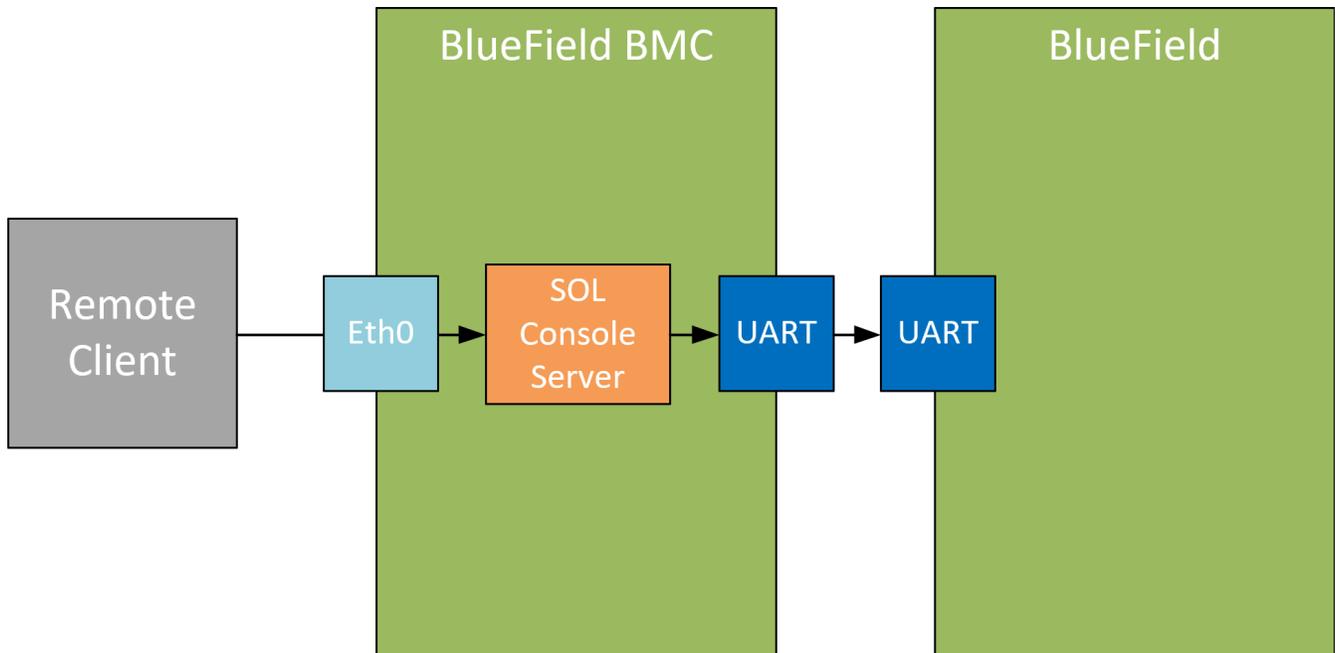
This value indicates if user configuration was accepted or ignored by the UEFI.

# Serial Over LAN

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": true  
    }  
  },  
  ...  
}
```

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

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

The port number can be obtained 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	<pre>ipmitool sol info ipmitool sol info 1</pre>	Get SOL configuration data
2	Enable SOL access	<pre>ipmitool sol set set-in-</pre>	Enable the properties to be set via set-in-progress then enable SOL access

No.	Function	Command	Description
		<pre>progress set-complete 1</pre> <pre>ipmitool sol set enabled true 1</pre>	
3	Activate SOL	<pre>ipmitool -C 17 -I lanplus -U &lt;username&gt; -P &lt;password&gt; -H &lt;ip_address&gt; sol activate</pre> <p>Where:</p> <ul style="list-style-type: none"> <li>• -U – BMC username</li> <li>• -H – BMC IP address</li> <li>• -P – BMC password</li> </ul>	Activate SOL access to the BlueField console
4	Deactivate SOL	<pre>ipmitool -C 17 -I lanplus -U &lt;username&gt; -P &lt;password&gt; -H &lt;ip_address&gt; sol deactivate</pre>	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-filestystems(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`).

---

# 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
```

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.

---

# 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: <pre>ipmitool raw 0x32 0x67 0x01</pre>
Disable VFM	Run from Arm/BlueField OS and reboot NIC-BMC: <pre>ipmitool raw 0x32 0x67 0x00</pre>

Operation Description	Command
Fetch VFM	<p>Run from Arm OS:</p> <pre data-bbox="789 264 1463 422">ipmitool raw 0x32 0x68</pre>
Get the status of the tunnel through UART	<p>Run the following command on the host where the BMC is connected:</p> <pre data-bbox="789 522 1463 680">echo -e "\\g\\@" &gt; /dev/ttyUSBX</pre> <p>Where <code>/dev/ttyUSBX</code> 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.</p>
Start tunneling on BMC through UART	<p>Run the following command on the host where the BMC is connected:</p> <pre data-bbox="789 1031 1463 1283">echo "touch /tmp/fw-update/uart-tunneling" &gt; /dev/ttyUSBX</pre> <p>Where <code>/dev/ttyUSBX</code> is the UART port number to which BMC is connected.</p>
Stop tunneling on BMC through UART	<p>Run the following command on the host where the BMC is connected:</p> <pre data-bbox="789 1478 1463 1635">echo -e "\r~." &gt; /dev/ttyUSBX</pre> <p>Where <code>/dev/ttyUSBX</code> is the UART port number to which BMC is connected.</p>
Reboot BMC through UART	<p>Run the following command on the host where the BMC is connected:</p>

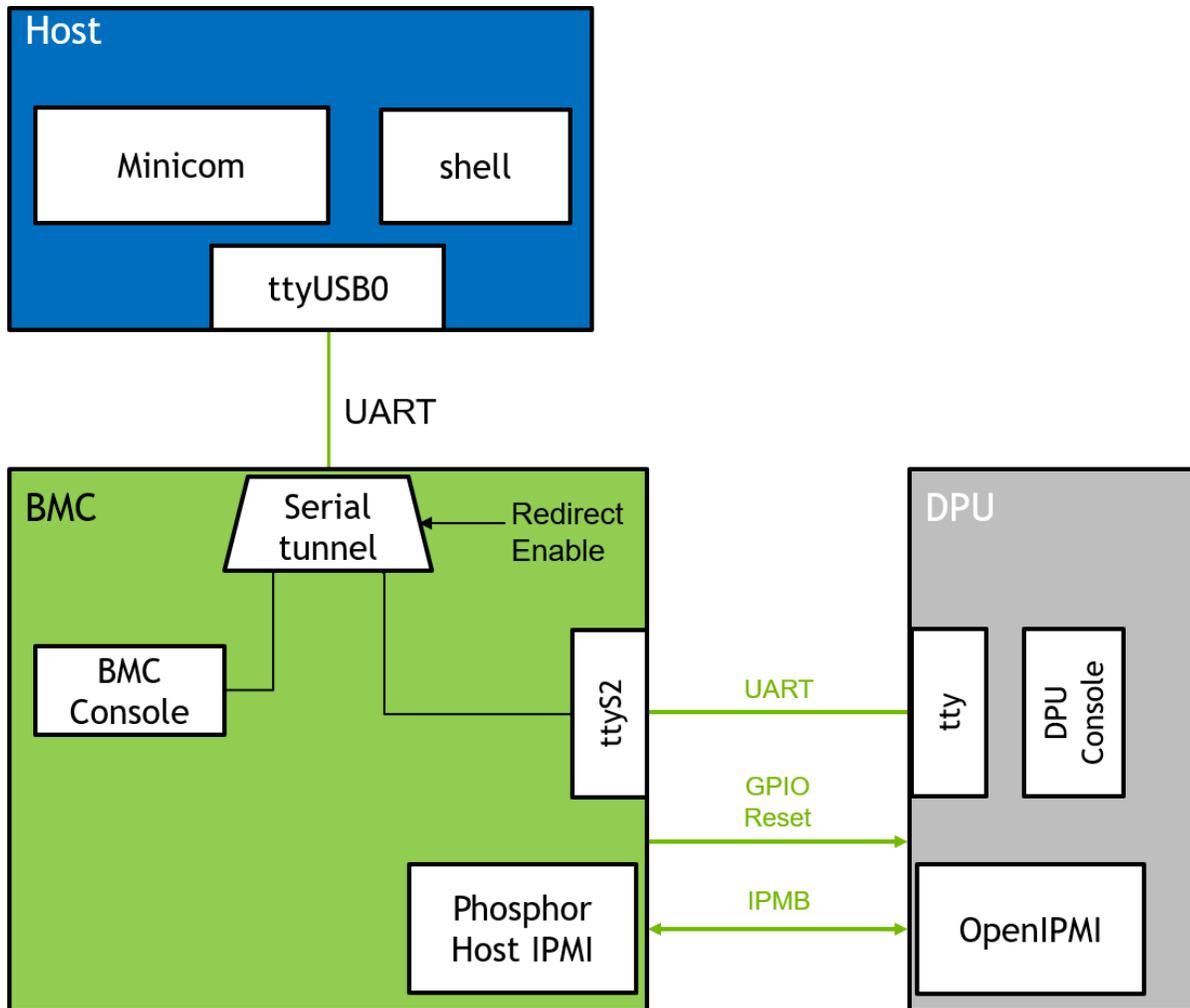
Operation Description	Command
	<pre data-bbox="789 212 1463 422">echo "touch /tmp/fw- update/reboot" &gt; /dev/ttyUSBX</pre> <p data-bbox="789 422 1463 520">Where <code>/dev/ttyUSBX</code> is the UART port number to which BMC is connected.</p>
<p data-bbox="155 709 683 783">To start/activate the BMC firmware update on BMC through UART</p>	<p data-bbox="789 531 1382 615">Run the following command on the host where the BMC is connected:</p> <pre data-bbox="789 615 1463 863">echo "touch /tmp/fw- update/fwactivate" &gt; /dev/ttyUSBX</pre> <p data-bbox="789 863 1463 968">Where <code>/dev/ttyUSBX</code> is the UART port number to which BMC is connected.</p>
<p data-bbox="155 1192 683 1266">To check the BMC firmware update status on BMC</p>	<p data-bbox="789 978 1390 1020">Run the following command on the BMC:</p> <pre data-bbox="789 1020 1463 1171">cat /tmp/fw-update/fwstatus</pre> <p data-bbox="789 1171 1463 1213">Output and their values:</p> <ul data-bbox="829 1251 1455 1444" style="list-style-type: none"> <li>• Activating – indicates firmware update is in progress</li> <li>• Active – indicates firmware update succeeded</li> <li>• Failed – indicates firmware update failed</li> </ul>
<p data-bbox="155 1497 667 1581">To check the CEC firmware update status on BMC</p> <div data-bbox="155 1633 764 1860" style="background-color: #ffffcc; padding: 10px;"> <p data-bbox="188 1675 513 1801"><b>i Note</b> Relevant only for BlueField-2.</p> </div>	<p data-bbox="789 1497 1390 1539">Run the following command on the BMC:</p> <pre data-bbox="789 1539 1463 1745">cat /tmp/cec_images progress.txt</pre> <p data-bbox="789 1745 1463 1787">Sample output of the <code>progress.txt</code>:</p> <ul data-bbox="829 1829 1219 1871" style="list-style-type: none"> <li>• CEC update in progress:</li> </ul>

Operation Description	Command
	<pre data-bbox="867 268 1458 468">TaskState="Running" TaskStatus="OK" TaskProgress="50"</pre> <ul data-bbox="829 474 1214 510" style="list-style-type: none"> <li>• CEC update completed:</li> </ul> <pre data-bbox="867 510 1458 814">TaskState=Firmware update succeeded. TaskStatus=OK TaskProgress=100</pre>
<p data-bbox="159 1163 651 1241">Transfer BMC firmware image for firmware update through UART</p>	<p data-bbox="787 871 1382 947">Run the following command on the host where the BMC is connected:</p> <pre data-bbox="787 947 1458 1150">echo -e -n "\ncd /tmp/images\n\nrz\n" &gt; /dev/ttyUSBX</pre> <p data-bbox="787 1157 1382 1232">Run the following command on the host where the BMC is connected:</p> <pre data-bbox="787 1232 1458 1436">sz -8b OTA.tar &lt; /dev/ttyUSBX &gt; /dev/ttyUSBX</pre> <p data-bbox="787 1442 1393 1528">Where <code>/dev/ttyUSBX</code> is the UART port number to which BMC is connected.</p>

Operation Description	Command
<p data-bbox="159 373 641 449">Transfer CEC firmware image for firmware update through UART</p> <div data-bbox="186 541 516 674" style="background-color: #ffffcc; padding: 10px;"> <p data-bbox="186 541 516 674"><b>i Note</b> Relevant only for BlueField-2.</p> </div>	<p data-bbox="787 226 1383 302">Run the following command on the host where the BMC is connected:</p> <div data-bbox="787 302 1463 558" style="border: 1px solid #ccc; padding: 10px;"> <pre data-bbox="846 365 1365 499">echo -e -n "\ncd /tmp/cec_images\n \nrz\n" &gt; /dev/ttyUSBX</pre> </div> <p data-bbox="787 562 1383 638">Run the following command on the host where the BMC is connected:</p> <div data-bbox="787 638 1463 846" style="border: 1px solid #ccc; padding: 10px;"> <pre data-bbox="846 701 1442 785">sz -8b OTA.bin &lt; /dev/ttyUSBX &gt; /dev/ttyUSBX</pre> </div> <p data-bbox="787 850 1395 934">Where <code>/dev/ttyUSBX</code> is the UART port number to which BMC is connected.</p>

# Serial Redirect Mode

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



To enable/disable serial redirect mode:

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

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

1. Disables vendor field mode if enabled.
2. Enables auto login (only on the serial port) for the root user. Root user can also log in using SSH through the OOB port.
3. Enables tunneling on BMC through UART by default.
4. 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.

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

1. Disables auto login (only on serial port) for the root user.
2. Disables tunneling on BMC through UART by default.

The following sections list the supported commands.

## Enabling Serial Redirect Mode to Run from Arm or BMC OS

```
ipmitool raw 0x32 0x6D 0x01
```

### Note

To enable serial redirect mode by default, set `BMC_SERIAL_REDIRECT_ENABLE=yes` in `bf.cfg`.

## Disabling Serial Redirect Mode Settings from Being Run on Arm or BMC OS

```
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.

# Bare-metal Reprovisioning

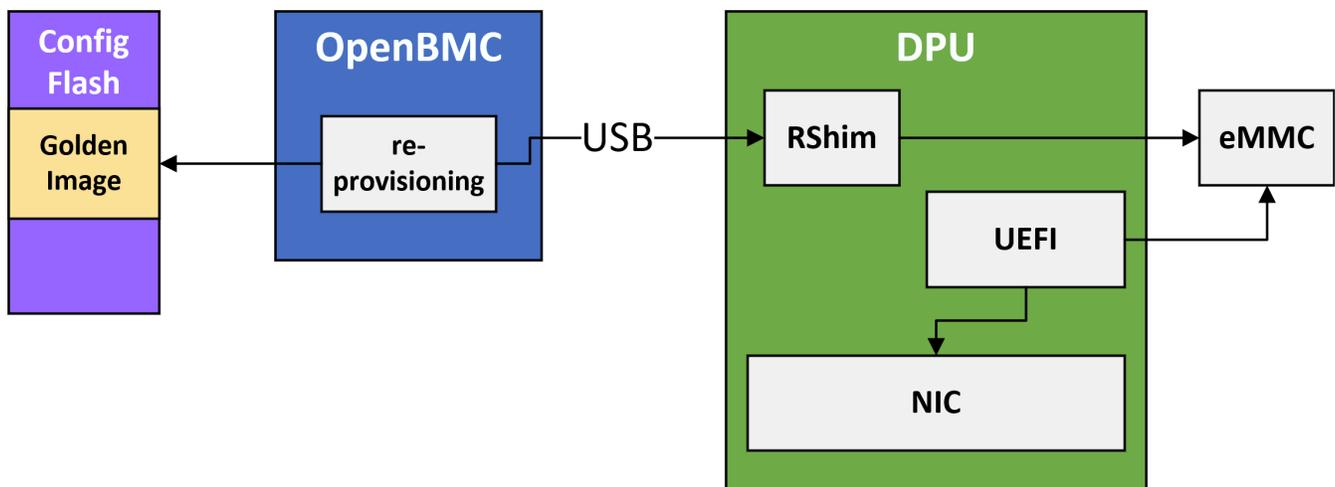
## **i** Note

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 network offers a solution for restoring the BlueField-3 system without relying on external measures. This method ensures the system can be brought back to its initial state, enabling the reloading of the operational image.

To facilitate this approach, the BMC is responsible for maintaining and managing a golden image for the UEFI and the NIC. This allows the UEFI to retrieve the operational image from the network via protocols such as HTTP or PXE.

The following block diagram describes in high level the system components and the data flow:



The entire flow of the network re-provisioning includes the following primary stages:

1. Initial provisioning of the golden images to the BMC.

## **i** Info

This process usually takes place during system manufacturing.

2. In-field update process enables the updating of golden images.
3. OOB network configuration involves configuring the network settings.
4. Recovering the system by reinstalling the golden images.

## **Initial Provisioning of Golden Image to BMC**

To initiate the initial provisioning of the Golden images, the BMC must be connected to the OOB network. The user is required to copy the images from their local storage to the BMC by utilizing a standard `scp` command over the network. Once the images are successfully located within the BMC, the user must log into the BMC to initiate the provisioning process which involves transferring the golden images into the BMC's non-volatile storage. To accomplish this, a dedicated utility provided within the BMC can be used. Users must ensure that the BMC remains powered on and uninterrupted during this stage to avoid potential problems.

The current flow supports the portioning of the golden images `golden_image_arm` and `golden_image_nic`.

To copy the golden images from the local environment into the BMC, run:

- 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/
```

### **i** Info

After the candidate image is copied into the BMC's volatile memory, the version is extracted from it and stored to support certain features.

### **i** Note

The NIC firmware version is extracted from the NIC firmware image filename, which only works if it is in the standard format of official releases.

After copying the golden images to the BMC's `/tmp/golden-image-nic` directory or `/tmp/golden-image-arm` directory, the user must log into the BMC and execute the following commands to provision the golden images into 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>
```

Once the golden images have been provisioned to the BMC's non-volatile storage, the user must execute the following commands to verify the correctness of the images:

- For `golden_image_nic`:

```
#bmc> dpu_golden_image -v golden_image_nic  
#bmc> echo $?
```

Expected output is 0.

- For `golden_image_arm`:

```
#bmc> dpu_golden_image -v golden_image_arm  
#bmc> echo $?
```

Expected output is 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, run:

- 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 values of the golden images, run:

- 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

1. To initiate an update, run the following command from the host:

- 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_  
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_  
https://<bmc-  
ip>/redfish/v1/UpdateService/Actions/UpdateService.SimpleU
```

Where:

- ImageURI – the image URI format should be  
`<remote-server-ip>/<golden-image-path>`
- bmc-ip – BMC IP address

After initiating the update, a new task is created for monitoring the progress:

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

2. To track the progress of the update:

```
curl -k -u root:'<password>' -X GET https://<bmc-  
ip>/redfish/v1/TaskService/Tasks/<task-id>
```

The update progress has three states: 0,10,100. After a successful update, the following output is expected:

```
"PercentComplete": 100,
```

```
"TaskState": "Completed",  
"TaskStatus": "OK"
```

In case of a failure, it is recommended to reboot the BMC and retry the update.

### Info

The golden image update may take between 1-3 minutes.

## 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 an IPMI command:

```
#bmc> ipmitool raw 0x32 0x99 <golden_image_timeout>  
<timeout_from_network> <verbosity_level> <halt_hard_reset>
```

This command is designed to be executed exclusively from within the BMC since it has a potentially disruptive impact on the system. When the command is executed, it extracts the golden images from the BlueField BMC's non-volatile memory and initiates the recovery process. Once the golden images are pushed to the RShim, the RShim console output is redirected to the BMC console, enabling the user to easily monitor the progress.

Upon successful completion of this command, both the BlueField NIC and Arm execute the designated GA image fetched from a preconfigured server.

- `golden_image_timeout` – timeout value, in minutes, for updating the golden images. For default value (15), users may input 0.
- `timeout_from_network` – timeout value, in minutes, for booting the operational image from the network. For default value (60), users may input 0.
- Verbosity level defines the type of messages that will appear during the reprovisioning process:
  - 0 – Quiet mode; only error messages appear on the screen
  - 1 – Info mode; only error messages and re-provisioning process messages appear on the screen
  - 2 – Full mode; all messages appear on the screen including BlueField RShim messages
- `halt_hard_reset` (optional) -s pecifies whether to halt the reprovisioning process before the final hard reset of the BlueField. This hard reset is the last step of the reprovisioning process and is necessary to activate the NIC firmware installed from the network.

Allowed Values:

- 0 - Perform the Hard reset to Complete the reprovisioning process (default behavior).
- 1 - Halt the reprovisioning process before performing the final hard reset of the BlueField.

### Info

Reprovisioning messages have the following prefix:  
`[<running date> GOLDEN-IMAGE-RECOVERY]`.

## Arm OS Signal to BlueField BMC When it Completes its Flow of Programming via RShim

After BFB installation is complete, the BlueField BMC waits for a specific sequence of messages over the RShim log:

```
NIC firmware update done
Installation finished
Linux up
```

- NIC firmware update done – This message indicates that the firmware update for the NIC subsystem has been successfully completed
- Installation finished – This message signals the completion of the installation process for the BFB from the network
- Linux up – Upon receiving this message, the BlueField BMC acknowledges 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.

## Notice

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

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

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

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

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

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

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

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

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

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

## **Trademarks**

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

Copyright 2024. PDF Generated on 12/16/2024