Miscellaneous

Table of contents

Bare-metal Reprovisioning 3
Power Capping 18
RShim Over USB 23
Serial Over LAN 26
Serial Redirect Mode 32
Vendor Field Mode 35

This section contains the following topics:

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

Bare-metal Reprovisioning

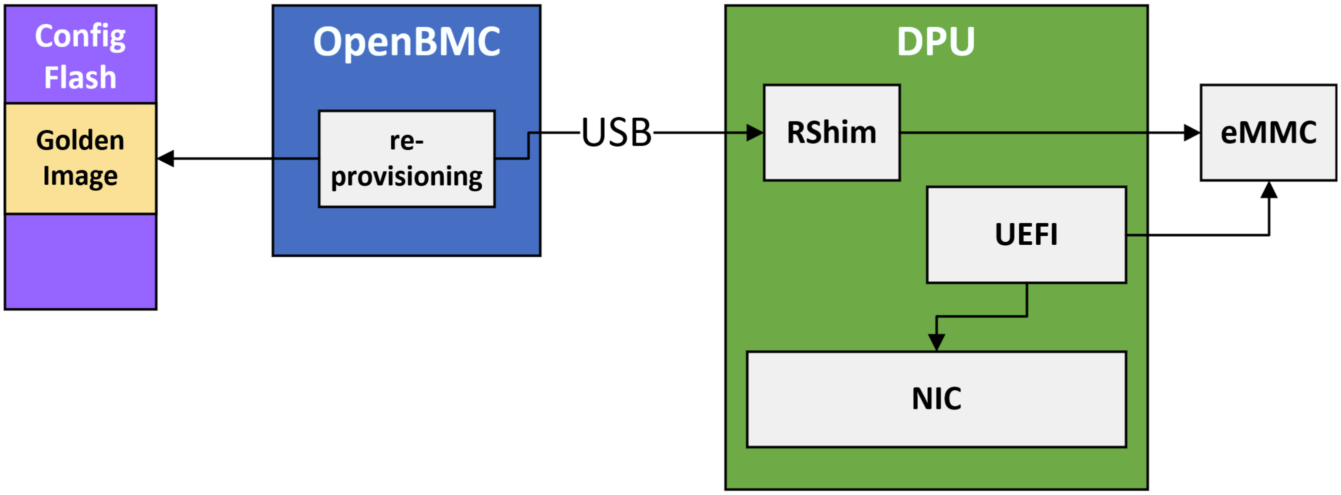
Note

Relevant for NVIDIA® BlueField®-3 and later in DPU mode only (not supported in NIC mode).

The re-provisioning flow of the BlueField-3 bare metal system offers a solution for restoring the system to its initial state using built-in resources, eliminating the need for external measures. This approach enables the seamless reloading of the operational image.

To support this functionality, the BMC maintains and manages a golden image for the UEFI and the NIC. This ensures that the UEFI can retrieve the operational image via network protocols such as HTTP or PXE.

The following block diagram provides a high-level overview of the system components and data flow:



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

1. Initial provisioning – Provisioning the golden images to the BMC, typically performed during system manufacturing.

2. In-field updates – Updating the golden images using the in-field update process.

3. OOB network configuration – Configuring network settings through out-of-band (OOB) management.

4. System recovery – Restoring the system by reinstalling the golden images.

Initial Provisioning of Golden Image to BMC

1. Ensure the BMC is connected to the out-of-band (OOB) network.

2. Use the `scp` command to transfer the golden images from your local storage to the BMC's temporary storage directories (`/tmp/golden-image-nic/` or `/tmp/golden-image-arm/`).

   - For `golden_image_nic`:

     ```
     #host> scp <nic-golden-image-directory>/<nic-golden-image-filename> root@<bmc-ip>:/tmp/golden-image-nic/
     ```

   - For `golden_image_arm`:

     ```
     #host> scp <arm-golden-image-directory>/<arm-golden-image-filename> root@<bmc-ip>:/tmp/golden-image-arm/
     ```

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

   > **Note**

The NIC firmware version is extracted from the image filename. Ensure the filename follows the standard format of official releases.

3. Log into the BMC and use the `dpu_golden_image` utility to transfer the golden images from temporary storage to the BMC's non-volatile storage.

   - For `golden_image_nic`:

     ```
     #bmc> dpu_golden_image golden_image_nic -w /tmp/golden-image-nic/<nic-golden-image-filename>
     ```

   - For `golden_image_arm`:

     ```
     #bmc> dpu_golden_image golden_image_arm -w /tmp/golden-image-arm/<arm-golden-image-filename>
     ```

   Info

   If the version of the candidate image matches the one already in non-volatile storage, the update is skipped, and the following message is displayed:

   ```
   Updating image in memory is cancelled as version will remain unchanged. Force update by adding -f|--force to the command line.
   ```

   To force the update, use the `--force` flag:

        - For `golden_image_nic`:

          ```
          #bmc> dpu_golden_image golden_image_nic -w /tmp/golden-image-nic/<nic-golden-image-filename> --force
          ```

        - For `golden_image_arm`:

          ```
          #bmc> dpu_golden_image golden_image_arm -w /tmp/golden-image-arm/<arm-golden-image-filename> --force
          ```

4. After provisioning, verify the correctness of the golden images using the following commands:

    - For `golden_image_nic`:

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

      Expected output: `0`.

    - For `golden_image_arm`:

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

      Expected output: `0`.

Retrieving Golden Image Version Information

Note

This feature is available only for golden images installed following the upgrade of the BMC firmware to version 24.07-14 or later.

To get the human-readable version (`MAJOR.MINOR.PATCH.BUILD` versioning scheme) of the golden images:

- For `golden_image_nic`:

```
bmc> dpu_golden_image golden_image_nic -V -H
```

- For `golden_image_arm`:

```
bmc> dpu_golden_image golden_image_arm -V -H
```

To get the `sha256sum` value:

- For `golden_image_nic`:

```
bmc> dpu_golden_image golden_image_nic -V
```

- For `golden_image_arm`:

```
bmc> dpu_golden_image golden_image_arm -V
```

Retrieving Golden Image Version Information Using Redfish

Note

This feature is available only for Golden Images installed following the upgrade of the BMC firmware to version 24.07-14 or later.

To get the human-readable version (`MAJOR.MINOR.PATCH.BUILD` versioning scheme) of the golden images over the Redfish interface, run:

- For Arm golden image:

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

- For NIC golden image:

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

Updating Golden Images Using Redfish

The process for updating golden images via Redfish includes the following steps.

1. The process for updating golden images via Redfish includes the following steps:
   - For NIC golden image:

     ```
     curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST
     -d '{"TransferProtocol":"HTTP", "ImageURI":"<remote-server-ip>/<nic-golden-image-path>",
     ```

```
"Targets":["redfish/v1/UpdateService/FirmwareInventory/golden_image_nic"]}'
https://<bmc-
ip>/redfish/v1/UpdateService/Actions/UpdateService.SimpleU
```

- For Arm golden image:

```
curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST
-d '{"TransferProtocol":"HTTP", "ImageURI":"<remote-server-ip>/<arm-golden-image-path>",
"Targets":["redfish/v1/UpdateService/FirmwareInventory/golden_image_arm"]}'
https://<bmc-
ip>/redfish/v1/UpdateService/Actions/UpdateService.SimpleU
```

Parameters:

- `ImageURI` – Specify the image location in the format `<remote-server-ip>/<golden-image-path>`
- `bmc-ip` – Specify the IP address of the BMC

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

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

2. Track the update's progress using the following command:

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

Update states:

- `0%` – Update started.
- `10%` – Update in progress.
- `100%` – Update complete.

  Expected output after a successful update:

  ```
  "PercentComplete": 100,
  "TaskState": "Completed",
  "TaskStatus": "OK"
  ```

  In case of failure, reboot the BMC and retry the update.

  > **Info**
  >
  > The golden image update process typically takes 1–3 minutes to complete.

If the candidate image version matches the one stored in the BMC's non-volatile memory, the update is skipped by default, and the following response is returned:

```
{
  ...
  {
      "@odata.type": "#MessageRegistry.v1_4_1.MessageRegistry",
```

```
      "Message": "The update operation for the component 'BMC' is
skipped because 'Component image is identical'.",
      "MessageArgs": [
        "BMC",
        "Component image is identical"
      ],
      "MessageId": "NvidiaUpdate.1.0.ComponentUpdateSkipped",
      "Resolution": "Retry firmware update operation with the
force flag",
      "Severity": "OK"
    },
  ],
  ...
  "TaskState": "Completed",
  "TaskStatus": "OK"
}
```

To override the default behavior and force the update, include the `"ForceUpdate": true` parameter in the command:

- For NIC golden image:

  ```
  curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST -d
  '{"TransferProtocol":"HTTP", "ImageURI":"<remote-server-ip>/<nic-golden-image-path>", "Targets":
  ["redfish/v1/UpdateService/FirmwareInventory/golden_image_nic"], "ForceUpdate": true}'
  https://<bmc-
  ip>/redfish/v1/UpdateService/Actions/UpdateService.SimpleUpdat
  ```

- For Arm golden image:

  ```
  curl -k -u root:'<password>' -H "Content-Type: application/json" -X POST -d
  '{"TransferProtocol":"HTTP", "ImageURI":"<remote-server-ip>/<arm-golden-image-path>", "Targets":
  ["redfish/v1/UpdateService/FirmwareInventory/golden_image_arm"], "ForceUpdate": true}'
  ```

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

OOB Network Configuration

To enhance the system's security, a new mechanism has been introduced to control network connectivity over the OOB network. This new feature provides an IPMI command to disable any communication between the BlueField BMC, BlueField, and the OOB management network. A set of IPMI commands are introduced to selectively enable the network on each of the above interfaces. This permits the platform's RoT to have complete control over which network interfaces can be enabled and when.

> **Note**
>
> This IPMI can only be sent by the platform's ROT. OOB and BlueField are blocked.

By default, the OOB interface is enabled. However, for the host BMC to gain control over this interface, it must disable it during the initial boot. Once disabled, the interface remains in that state regardless of BMC reboots or system cold boots.

For more details, refer to "OOB Network 3-Port Switch Control".

Golden Images Reprovisioning

The re-provisioning flow is initiated using the following IPMI command:

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

This command must be executed from within the BMC, as it can significantly impact the system. Upon execution:

1. The golden images are extracted from the BlueField BMC's non-volatile memory.

2. The recovery process is initiated, pushing the golden images to the RShim.
3. The RShim console output is redirected to the BMC console, allowing the user to monitor the process.

Once the process completes, both the BlueField NIC and ARM execute the designated golden images retrieved from a preconfigured server.

Command parameters:

- `<golden_image_timeout>` – Timeout value (in minutes) for updating the golden images. Default Value: `15` minutes (input `0` to use the default).
- `<timeout_from_network>` – Timeout value (in minutes) for booting the operational image from the network. Default Value: `60` minutes (input `0` to use the default).
- `<verbosity_level>` – Controls the level of detail displayed during the reprovisioning process:
    - `0` – Quiet mode (only error messages are displayed)
    - `1` – Info mode (error messages and reprovisioning process messages are displayed)
    - `2` – Full mode (all messages, including BlueField RShim messages, are displayed)
- `<halt_hard_reset>` (Optional) – Specifies whether to halt the reprovisioning process before performing the final hard reset of the BlueField.

    Note

    The final hard reset is crucial to activate the NIC firmware installed from the network.

    - `0` – Perform the hard reset to complete the reprovisioning process (default)

    - `1` – Halt the process before the final hard reset

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

Signaling BlueField BMC After Arm OS Programming Completion

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

```
NIC firmware update done        # Indicates that the firmware update for the NIC subsystem has been successfully completed
Installation finished           # Signals the completion of the installation process for the BFB from the network
Linux up                                        # Indicates that the BlueField BMC has acknowledged that the Arm OS has booted up and is ready
```

BlueField BMC expects these messages in the specified order.

Adding Entries to RShim Log from BlueField Arm OS

Users can add custom entries to the RShim log from the BlueField Arm OS using the `bfrshlog` command. The syntax of the command is: `bfrshlog <output>`.

For example, to add the message "Linux up" to the RShim log, run:

```
bfrshlog "Linux up"
```

Expected Output

- All output from the BlueField Arm console is redirected to the BlueField BMC console for monitoring purposes.
- The steps of the re-provisioning process are printed with `[<running date> GOLDEN-IMAGE-RECOVERY]` prefix and are outlined in the following:

```
[<running date> GOLDEN-IMAGE-RECOVERY]  Checking pcie slot is in reset
[<running date> GOLDEN-IMAGE-RECOVERY]  Read golden images from flash
[<running date> GOLDEN-IMAGE-RECOVERY] Set FNP to 0
[<running date> GOLDEN-IMAGE-RECOVERY] Checking rshim interface after SOC hard reset
[<running date> GOLDEN-IMAGE-RECOVERY]  Starting ATF/UEFI golden image update
[<running date> GOLDEN-IMAGE-RECOVERY] Finished updating ATF/UEFI golden image
[<running date> GOLDEN-IMAGE-RECOVERY] Starting NIC FW golden image update
[<running date> GOLDEN-IMAGE-RECOVERY] Finished updating NIC FW golden image
[<running date> GOLDEN-IMAGE-RECOVERY] Stop Redfish server
[<running date> GOLDEN-IMAGE-RECOVERY] Configure Recovery image to boot from network
[<running date> GOLDEN-IMAGE-RECOVERY] set FNP to 1
[<running date> GOLDEN-IMAGE-RECOVERY] Booting BFB from network
[<running date> GOLDEN-IMAGE-RECOVERY] Start Redfish server
[<running date> GOLDEN-IMAGE-RECOVERY] Set boot option to default
if halt_hard_reset is 0:
[<running date> GOLDEN-IMAGE-RECOVERY] Finished programming image from network. Start DPU hard reset
if halt_hard_reset is 1:
```

```
[<running date> GOLDEN-IMAGE-RECOVERY] Finished programming image from network
[<running date> GOLDEN-IMAGE-RECOVERY] The Reprovisioning process was halted at user's request. To complete the process, please power cycle the device
```

A failed update prints the following:

```
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: aborting process! PCIE is not in reset.
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: Reading golden_image_nic failed
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: Reading golden_image_arm failed
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: rshim has not started successfully
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: pushing ATF/UEFI golden image over rshim failed
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: programming of ATF/UEFI golden image failed
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: pushing NIC FW golden image over rshim failed
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: programming of NIC FW golden image failed
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: failed to configure image to boot from network
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR:  programming of image from network failed: NIC firmware update failed
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: programming of image from network failed: Installation failed
[<running date> GOLDEN-IMAGE-RECOVERY] ERROR: programming of image from network failed: Failed to get Linux up
```

Due to line buffering in the BlueField Arm console, buffered output lines receive the same timestamp value in `<running date>` when they are redirected to the BlueField BMC console.

Power Capping

Info

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

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

Redfish Power Capping Requests

Getting General Power Capping Information

Control information:

```
curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/Chassis/Card1/Controls/PowerLimit_0
```

Output example:

```
{
  "@odata.id": "/redfish/v1/Chassis/Card1/Controls/PowerLimit_0",
  "@odata.type": "#Control.v1_3_0.Control",
  "AllowableMax": 300,
  "AllowableMin": 200,
  "ControlMode": "Manual",
  "ControlType": "Power",
  "Id": "PowerLimit_0",
  "Name": "System Power Control",
  "SetPoint": 10,
  "SetPointUnits": "%",
  "Status": {
    "Health": "OK",
    "HealthRollup": "OK",
    "State": "Enabled"
  }
}
```

Enabling/Disabling Adjustment of Power Capping

```
curl -k -u root:'<password>' -H "Content-Type: application/json"
-X PATCH
https://<bmc_ip>/redfish/v1/Chassis/Card1/Controls/PowerLimit_0 -
d '{"ControlMode":<"Manual"/"Disabled">}'
```

Info

The ability to adjust power capping is disabled by default.

Setting Power Allocation Percentage

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

```
https://<bmc_ip>/redfish/v1/Chassis/Card1/Controls/PowerLimit_0 -
d '{"SetPoint": <val>}'
```

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

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

IPMI Power Capping Commands

Getting Power Capping Status

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

Enabling/Disabling Adjustment of Power Capping

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

Where `val`:

- 0 – disable
- 1 – enable

Note

Changeable only from BMC prompt using `admin` account.

Info

The ability to adjust power capping is disabled by default.

Getting Power Capping Percentage

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

Setting Power Capping Percentage

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

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

Note

Changeable only from BMC prompt using `admin` account.

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

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

Getting Maximum Power Capacity

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

ⓘ **Info**

Power is given in watts.

Getting Minimum Power Capacity

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

ⓘ **Info**

Power is given in watts.

RShim Over USB

Network Connection from BMC to BlueField

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

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

Enabling RShim on BlueField BMC

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

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

ⓘ **Info**

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

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

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

Note

RShim can be force enabled on BMC even it is started on host.
The previous steps are not necessary in this case:

```
curl -k -u root:'<password>' -H "Content-
Type: application/json" -X POST -d '{"Rshim":
"Forced"}'
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Actior
```

0000019a-a294-d3f6-abda-b29f6aff0003

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

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

Info

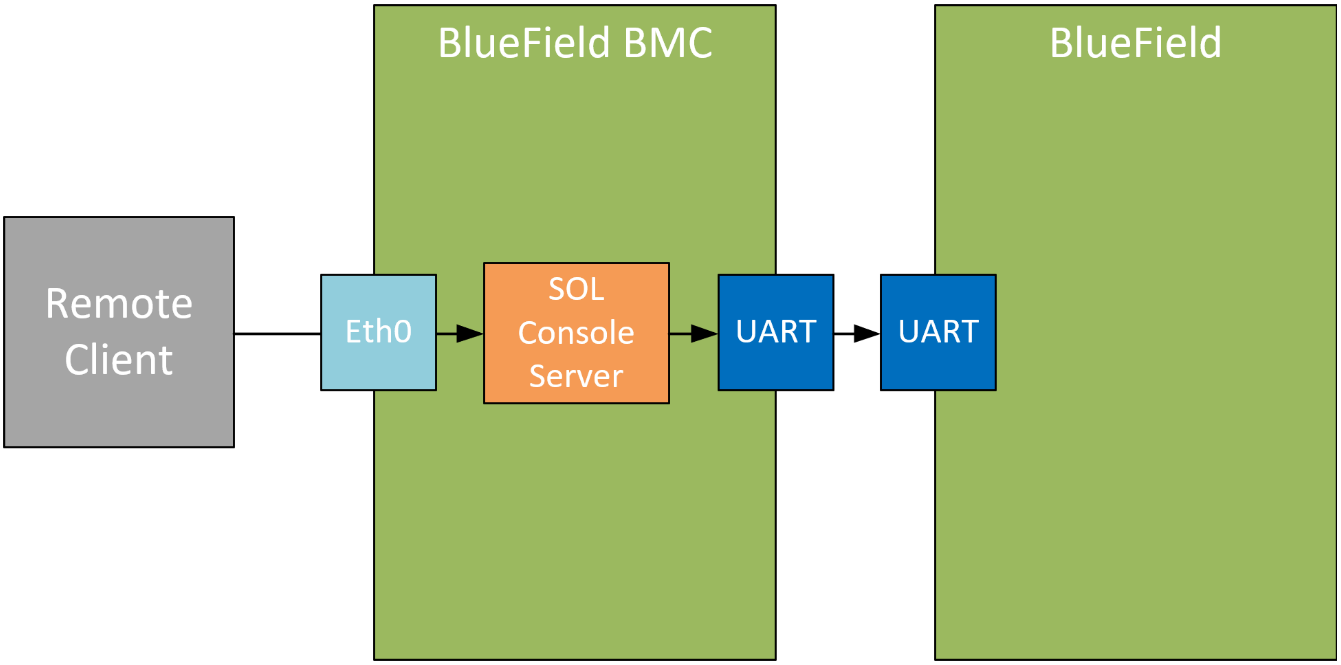
This may take up to 8 seconds. If the `BmcRShimEnabled` value does not change, disable BMC RShim by setting the value to `false` then repeating steps 1-3.

Serial Over LAN

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

ⓘ **Note**

The serial-over-LAN (SOL) connection is first established using the BlueField BMC credentials. Once the connection to the BlueField BMC is successful, the serial communication is redirected to the BlueField Arm console, where additional BlueField Arm credentials are required to complete the connection.



SOL Redfish Commands

To establish the SOL connection, users may retrieve information from the `redfish/v1/Systems/Bluefield` schema. Inside the `SerialConsole` properties

(SSH, IPMI), there are various methods that a client can utilize to initiate a serial session with the host through its manager.

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

Example output:

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

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

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

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

SOL IPMI Commands

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

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

For example:

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

Poky (Yocto Project Reference Distro)

2.3.1 bluefield /dev/ttyAMA0

bluefield login:
```

The IPMI SOL commands are listed in the following table:

No.	Function	Command	Description

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

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

No.	Function	Command	Description
		`ipmitool sol set enabled true 1`	
3	Activate SOL	`ipmitool -C 17 -I lanplus -U <username> -P <password> -H <ip_address> sol activate` Where: - -U – BMC username - -H – BMC IP address - -P – BMC password	Activate SOL access to the BlueField console
4	Deactivate SOL	`ipmitool -C 17 -I lanplus -U <username> -P <password> -H <ip_address> sol deactivate`	Deactivate SOL access to the BlueField console

Note

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

SysRq Support in SOL

SysRq is a special key combination used by Linux to perform various low-level commands. SOL invokes the SysRq feature by sending a serial break signal, followed by the desired key. To enable SysRq, the user must issue the following command on the BlueField:

```
echo 1 > /proc/sys/kernel/sysrq
```

In the SOL of BMC, the break signal is generated by keys `\n~B`. So, in an SOL session, the user may enter the `\n~B` keys to trigger the break and then enter a keycode as the SysRq command.

For example, the following are the outputs when inputting `h` after generating a break signal in the SOL session:

- For SOL over IPMI:

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

  Poky (Yocto Project Reference Distro)

  2.3.1 bluefield /dev/ttyAMA0

  bluefield login:
  bluefield login: ~B [send break]
  [490472.371785] sysrq: HELP : loglevel(0-9) reboot(b) crash(c) terminate-all-tasks(e) memory-full-oom-kill(f) kill-all-tasks(i) thaw-filesystems(j) sak(k) show-backtrace-all-active-cpus(l) show-memory-usage(m) nice-all-RT-tasks(n) poweroff(o) show-registers(p) show-all-timers(q) unraw(r) sync(s) show-task-states(t) unmount(u) show-blocked-tasks(w) dump-ftrace-buffer(z)
  ```

- For SOL over SSH (break signal generated with \n~~B):

  ```
  ssh -p 2200 root@10.10.10.10
  ```

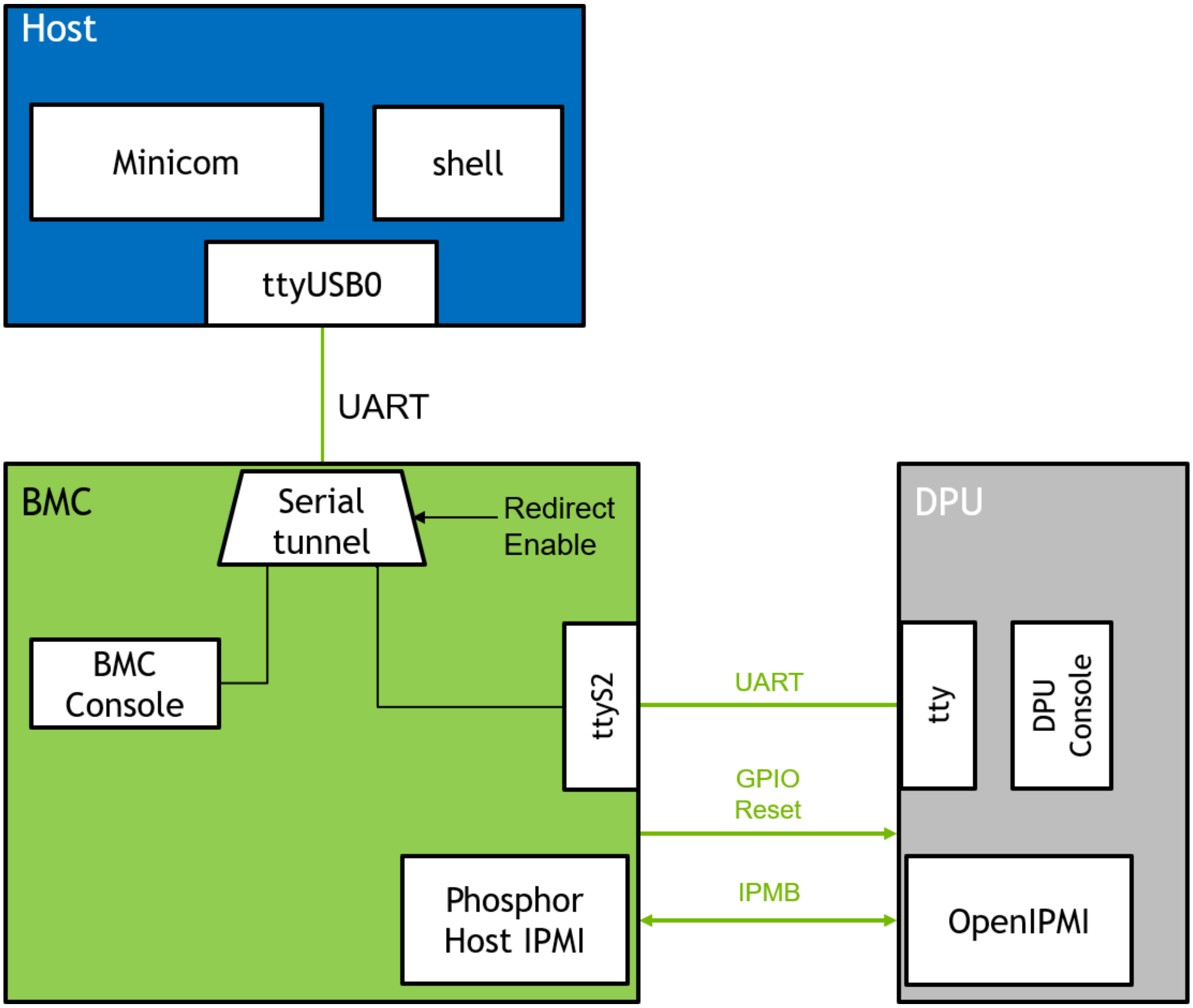
```
root@10.10.10.10's password:

bluefield login:
[490472.371785] sysrq: HELP : loglevel(0-9) reboot(b)
crash(c) terminate-all-tasks(e) memory-full-oom-kill(f) kill-
all-tasks(i) thaw-filesystems(j) sak(k) show-backtrace-all-
active-cpus(l) show-memory-usage(m) nice-all-RT-tasks(n)
poweroff(o) show-registers(p) show-all-timers(q) unraw(r)
sync(s) show-task-states(t) unmount(u) show-blocked-tasks(w)
dump-ftrace-buffer(z)
```

Note

In the context of SOL over SSH connections, an additional tilde symbol ('`~`') is used to navigate through multiple layers of SSH sessions to access the SOL session (e.g., `\n~~B`).

Serial Redirect Mode

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



To enable/disable serial redirect mode:

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

The following sections list the supported commands.

Enabling Serial Redirect Mode to Run from Arm or BMC OS

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

1. Disables vendor field mode if enabled.
2. Enables tunneling on BMC through UART by default.
3. BlueField BMC validates that BlueField is in controller mode (refer to the self-hosted SKUs), and if so, it resets (`SOC_HARD_RESET`) the BlueField.

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

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

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

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

```
ipmitool raw 0x32 0x6D 0x00
```

Fetching Serial Redirect Mode Settings

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

Starting Tunneling on BMC Through UART

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

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

Stopping Tunneling on BMC Through UART

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

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

Where `/dev/ttyUSBX` is the UART port number to which the BMC is connected.

User can also stop tunneling by running the command `~.` on the console client.

ⓘ **Info**

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

Vendor Field Mode

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

Enabling VFM automatically performs the following on BMC:

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

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

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

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

Disabling VFM automatically performs the following on BMC:

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

Updating BMC Firmware with Vendor Field Mode

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

8. Check the new BMC firmware version.

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

Updating CEC Firmware with Vendor Field Mode

> **Note**
>
> Relevant only for BlueField-2.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Updating BMC and Glacier Firmware with Vendor Field Mode

Note

Relevant only for BlueField-3.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

8. Check the new BMC firmware version.

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

Supported Vendor Field Mode Commands

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

Enable VFM:

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

Disable VFM:

```
ipmitool raw 0x32 0x67 0x00
```

Operation Description	**Command**
Fetch VFM	Run from Arm OS: `ipmitool raw 0x32 0x68`
Get the status of the tunnel through UART	Run the following command on the host where the BMC is connected: `echo -e "\\g\\@" > /dev/ttyUSBX` Where `/dev/ttyUSBX` is the UART port number to which BMC is connected. Expect the following sequence of chars when the tunnel is up and running: 169 150 230. Expect the following sequence of chars when the tunnel is not running: 165 200.
Start tunneling on BMC through UART	Run the following command on the host where the BMC is connected: `echo "touch /tmp/fw-update/uart-tunneling" > /dev/ttyUSBX` Where `/dev/ttyUSBX` is the UART port number to which BMC is connected.
Stop tunneling on BMC through UART	Run the following command on the host where the BMC is connected: `echo -e "\r~." > /dev/ttyUSBX` Where `/dev/ttyUSBX` is the UART port number to which BMC is connected.
Reboot BMC through UART	Run the following command on the host where the BMC is connected: `echo "touch /tmp/fw-`

Operation Description	**Command**
	`update/reboot" > /dev/ttyUSBX` Where `/dev/ttyUSBX` is the UART port number to which BMC is connected.
To start/activate the BMC firmware update on BMC through UART	Run the following command on the host where the BMC is connected: `echo "touch /tmp/fw-update/fwactivate" > /dev/ttyUSBX` Where `/dev/ttyUSBX` is the UART port number to which BMC is connected.
To check the BMC firmware update status on BMC	Run the following command on the BMC: `cat /tmp/fw-update/fwstatus` Output and their values: • Activating – indicates firmware update is in progress • Active – indicates firmware update succeeded • Failed – indicates firmware update failed

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

Operation Description	Command

	- CEC update completed:
	`TaskState=Firmware update succeeded.`
	`TaskStatus=OK`
	`TaskProgress=100`

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

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

Operation Description	Command
	Where `/dev/ttyUSBX` is the UART port number to which BMC is connected.

Notice

Trademarks