



CEC and BMC Firmware Operations

Table of contents

CEC and BMC Firmware Commands

Resetting/Rebooting BMC

Triggering Secure Firmware Update

Getting the Running CEC Firmware Version

Tracking Secure Firmware Update Progress

Getting the Running BMC Firmware Version

ForceUpdate

Updating BMC

Updating CEC

Activating New CEC

IPMI Command

Resetting CEC and BMC Subsystems Using IPMI I2C Command over SMBus Channel Connected to PCIe Golden Finger

Resetting CEC and BMC Subsystems Using CEC Self-reset Command

Resetting the Entire BlueField

Log Event Entries Created per Response Type

Redfish Command

CEC Background Update Status

Possible Error Codes

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



CEC and BMC Firmware Commands

Triggering Secure Firmware Update

i Info

Required for BMC/CEC update.

This section describes how to trigger a secure update and to track the secure update progress.

- To perform a multipart update with `MultipartHttpPushUri` API, run:

```
curl -k -u root:'<password>'
https://<bmc_ip>/redfish/v1/UpdateService/update-multipart -F
'UpdateParameters={};type=application/octet-stream' -F
UpdateFile=@<package_path>
```

- Update with `HttpPushUri` API – Deprecated

Warning

`HttpPushUri` API is obsolete. NVIDIA recommends users migrate to `MultipartHttpPushUri` API as support for `HttpPushUri` API will be discontinued in the future.

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

Where:

- `password` – password of root user
- `bmc_ip` – BMC IP address
- `package_path` – firmware update package path

Tracking Secure Firmware Update Progress

Info

Required for BMC/CEC update.

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

Find the current task ID in the response and use it for checking the progress by running:

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

Where:

- `password` – password of root user
- `bmc_ip` – BMC IP address
- `task_id` – Task ID

To retrieve additional information about the task progress, run:

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

The task status can be one of the following:

- `"Running"` – the update is currently in progress.
- `"Completed"` – the update finished successfully.
- `"Exception"` – an error occurred during the update.

Resetting/Rebooting BMC

Info

Required for BMC update.

To reset/reboot BMC, run:

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

Where:

- `password` – password of root user
- `bmc_ip` – BMC IP address

Getting the Running BMC Firmware Version

Info

Required for BMC update.

The following commands get the running firmware version of BlueField devices via the BMC.

- For NVIDIA® BlueField®-3:

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

Where:

- `password` – password of root user
- `bmc_ip` – BMC IP address
- For NVIDIA® BlueField®-2:

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

Get the current firmware ID and then perform:

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

Where:

- `password` – password of root user
- `bmc_ip` – BMC IP address
- `firmware_id` – numeric value found in the `FwInventory` schema only. It is calculated during firmware update by the BMC and used to distinguish between the versions.

Getting the Running CEC Firmware Version

Info

Required for CEC update.

Gets the running firmware version from CEC.

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

Where:

- `password` – password of root user
- `bmc_ip` – BMC IP address

ForceUpdate

Info

Relevant for BlueField-3 only.

i Info

Required for BMC update.

`ForceUpdate` forces the update procedure to proceed even if the target version is identical to the currently programmed version.

- When using `HttpPushUri` API, `ForceUpdate` is always `true` (i.e., forces the update procedure to proceed even if the target version is identical)
- When using `MultipartHttpPushUri` API, `ForceUpdate` is `false` (i.e., the update procedure fails if the target version is identical), unless `"ForceUpdate":true` is included under `UpdateParameters`:

```
curl -k -u root:'<password>'
https://<bmc_ip>/redfish/v1/UpdateService/update-multipart -F
'UpdateParameters=
{"ForceUpdate":true};type=application/octet-stream' -F
UpdateFile=@<package_path>
```

Where:

- - `password` – password of root user
 - `bmc_ip` – BMC IP address

Updating BMC

Info

Firmware update takes about 12 minutes.

After initiating the BMC secure update, you can expect to receive responses similar to the following.

- If an `HttpPushUri` API is used:

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

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

- If a `MultipartHttpPushUri` API is used:

```
curl -k -u root:'<password>'
https://<bmc_ip>/redfish/v1/UpdateService/update-multipart -F
'UpdateParameters={};type=application/octet-stream' -F
UpdateFile=@<package_path>
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/0",
  "@odata.type": "#Task.v1_4_3.Task",
  "Id": "<id>",
  "TaskState": "Running",
  "TaskStatus": "OK"
}
```

Where:

- - `package_path` – the BMC firmware image file including its path

For example:

```
curl -k -u root:'myP@ssword_12345!'
https://10.10.10.10/redfish/v1/UpdateService/update-multipart -F 'UpdateParameters=
{};type=application/octet-stream' -F UpdateFile=@/root/bf2-bmc-ota-24.01-4-ipn.tar
```

The following command is used to track secure firmware update progress:

```

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

  % Total    % Received % Xferd  Average Speed   Time    Time
Time  Current Dload  Upload    Total   Spent    Left   Speed
100 2123 100 2123    0     0 38600      0 --:--:-- --:--:-- -
-:--:-- 37910
20

```

The task is completed when `PercentComplete` reaches 100.

Since the reboot option is disabled during the update procedure, use the following command to reboot the BMC.

```

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

curl -k -u root:'<password>' -H "Content-Type: application/octet-
stream" -X POST -d '{"ResetType": "GracefulRestart"}'
https://<bmc_ip>/redfish/v1/Managers/Bluefield_BMC/Actions/Manager
{
  "@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The request completed successfully.",
      "MessageArgs": [],
      "MessageId": "Base.1.13.0.Success",
      "MessageSeverity": "OK",
      "Resolution": "None"
    }
  ]
}

```

The following commands are used to verify the current BMC firmware version after reboot:

- For BlueField-3:

```

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

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

```

- For BlueField-2:

1. Get the firmware ID from `FirmwareInventory`.

```

curl -k -u root:'<password>' -X GET
https://<bmc_ip>/redfish/v1/UpdateService/FirmwareInventory
{
  "@odata.id" :
  "/redfish/v1/UpdateService/FirmwareInventory",
  "@odata.type" :
  "#SoftwareInventoryCollection.SoftwareInventoryCollection"
  "Members": [
    {
      "@odata.id" :
      "/redfish/v1/UpdateService/FirmwareInventory/8c8549f3_BMC_
...

```

2. Use the following command with the retrieved firmware ID from the previous step.

```

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

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

```

For BlueField-3 BMC only: When updating firmware to a target that has an identical version using `MultipartHttpPushUri`, you must include `ForceUpdate=true`. Otherwise, the update procedure fails with the message `Component image is identical`.

```

curl -k -u root:'<password>'
https://<bmc_ip>/redfish/v1/UpdateService/update-multipart -F
'UpdateParameters={"ForceUpdate":true};type=application/octet-
stream' -F UpdateFile=@<package_path>

```

Updating CEC

Info

Firmware update takes about 20 seconds.

After initiating the BMC secure update, expect responses similar to the following-- depending whether `HttpPushUri` or `MultipartHttpPushUri` is used:

- `HttpPushUri` API:

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

- `MultipartHttpPushUri` API:

```
curl -k -u root:'<password>'
https://<bmc_ip>/redfish/v1/UpdateService/update-multipart -F
'UpdateParameters={};type=application/octet-stream' -F
UpdateFile=@<package_path>
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/0",
  "@odata.type": "#Task.v1_4_3.Task",
  "Id": "0",
  "TaskState": "Running",
  "TaskStatus": "OK"
}
```

Where:

- `package_path` – the BMC firmware image file including its path

For example:

```
curl -k -u root:'myP@ssword_12345!'
https://10.10.10.10/redfish/v1/UpdateService/update-
multipart -F 'UpdateParameters=
{};type=application/octet-stream' -F
UpdateFile=@/root/cec_ota_BMGP-04.0f_debug.bin
```

Use the following command to track the progress of the CEC firmware update.

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

Note

After the CEC secure update operation is complete, CEC activation and reset should be done (see "[Activating New CEC](#)") to apply the changes once the update is finished.

Use the following command to verify the current CEC firmware version after reboot.

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

```

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

Activating New CEC

Note

This is relevant only for BlueField-3 networking platforms (DPU or SuperNIC).

To activate the new CEC firmware, you must reset the CEC device. The following subsections describe possible reset options.

Resetting CEC and BMC Subsystems Using CEC Self-reset Command

Note

The CEC self-reset command is supported in CEC versions `00.02.0180.0000` and above. The command activates the new firmware on CEC and should only be used after the CEC update procedure is complete.

Redfish Command

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

Where:

- `password` – password of root user
- `bmc_ip` – BMC IP address

Command response options:

- The response in case of success:

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

- The response if there is no pending CEC firmware:

```

curl -k -u root:'<password>' -H "Content-Type:
application/json" -X POST
https://<bmc_ip>/redfish/v1/Chassis/Bluefield_ERoT/Actions/Cha
-d '{"ResetType":"GracefulRestart"}'
{
  "error": {
    "@Message.ExtendedInfo": [
      {
        "@odata.type": "#Message.v1_1_1.Message",
        "Message": "The requested resource of type ERoT FW
named 'Pending-ERoT-FW' was not found.",
        "MessageArgs": [
          "ERoT FW",
          "Pending-ERoT-FW"
        ],
        "MessageId": "Base.1.15.0.ResourceNotFound",
        "MessageSeverity": "Critical",
        "Resolution": "Provide a valid resource identifier
and resubmit the request."
      }
    ],
    "code": "Base.1.15.0.ResourceNotFound",
    "message": "The requested resource of type ERoT FW named
'Pending-ERoT-FW' was not found."
  }
}

```

- The response if the command is not supported by the current CEC firmware:

```
curl -k -u root:'<password>' -H "Content-Type:
application/json" -X POST
https://<bmc_ip>/redfish/v1/Chassis/Bluefield_ERoT/Actions/Cha
-d '{"ResetType":"GracefulRestart"}'
{
  "error": {
    "@Message.ExtendedInfo": [
      {
        "@odata.type": "#Message.v1_1_1.Message",
        "Message": "The action ERoT self-reset is not
supported by the resource.",
        "MessageArgs": [
          "ERoT self-reset"
        ],
        "MessageId": "Base.1.15.0.ActionNotSupported",
        "MessageSeverity": "Critical",
        "Resolution": "The action supplied cannot be
resubmitted to the implementation. Perhaps the action was
invalid, the wrong resource was the target or the
implementation documentation may be of assistance."
      }
    ],
    "code": "Base.1.15.0.ActionNotSupported",
    "message": "The action ERoT self-reset is not supported
by the resource."
  }
}
```

IPMI Command

```
ipmitool raw 0x32 0xD2
```

Command response options:

- If successful, no response is given. Glacier and BMC are reset.
- The response if there is no pending CEC firmware is:

```
Unable to send RAW command (channel=0x0 netfn=0x32 lun=0x0  
cmd=0xd2 rsp=0xd6): Cannot execute command, command disabled
```

- The response if the command is not supported by the current CEC firmware is:

```
Unable to send RAW command (channel=0x0 netfn=0x32 lun=0x0  
cmd=0xd2 rsp=0xd5): Command not supported in present state
```

Log Event Entries Created per Response Type

- Log event entry created in case of success:

```

{
  "@odata.id" :
"/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>"
  "@odata.type" : "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI" :
"/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>"
  "Created" : "<Date>",
  "EntryType" : "Event",
  "Id" : "<Id>",
  "Message" : "The request completed successfully.",
  "MessageArgs" : [
    ""
  ],
  "MessageId" : "Base.1.0.Success",
  "Name" : "System Event Log Entry",
  "Resolution" : "Ready for ERoT self Reset",
  "Resolved" : false,
  "Severity" : "OK"
}

```

- Log event entry created if there is no pending CEC firmware:

```

curl -k -u root:'<password>' -H "Content-Type:
application/json" -X GET
https://<bmc_ip>/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>
{
  "@odata.id":
"/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI":
"/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>",
  "Created": "<Date>",
  "EntryType": "Event",
  "Id": "<Id>",
  "Message": "Awaiting for an action to proceed with
installing image '' on 'ERoT'.",
  "MessageArgs": [
    "",
    "ERoT"
  ],
  "MessageId": "Update.1.0.AwaitToUpdate",
  "Name": "System Event Log Entry",
  "Resolution": "Cannot perform ERoT self reset: There is no
EC FW pending. Perform an ERoT FW update.",
  "Resolved": false,
  "Severity": "OK"
}

```

- Log event entry created if the command is not supported by the current CEC firmware:

```

{
  "@odata.id":
  "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>",
  "@odata.type": "#LogEntry.v1_13_0.LogEntry",
  "AdditionalDataURI":
  "/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<Id>",
  "Created": "<Date>",
  "EntryType": "Event",
  "Id": "<Id>",
  "Message": "The action ERoT self Reset is not supported by
the resource.",
  "MessageArgs": [
    "ERoT self Reset"
  ],
  "MessageId": "Base.1.0.ActionNotSupported",
  "Name": "System Event Log Entry",
  "Resolution": "Cannot perform ERoT self reset: The action
is not supported by the current ERoT version.",
  "Resolved": false,
  "Severity": "OK"
}

```

- Possible log event entries created if an error occurs during the BMC shutdown procedure (received after a success log event entry):

```
{
  "@odata.id":
"/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<I
...
  "MessageId": "Base.1.0.InternalError",
  "Name": "System Event Log Entry",
  "Resolution": "ERoT Reset: Failed to close services towards
ERoT reset",
  "Resolved": false,
  "Severity": "Critical"
}
```

```
{
  "@odata.id":
"/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<I
...
  "MessageId": "Base.1.0.InternalError",
  "Name": "System Event Log Entry",
  "Resolution": "ERoT Reset: Isolate operation may still be
in progress.",
  "Resolved": false,
  "Severity": "Critical"
}
```

```
{
  "@odata.id":
"/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<I
...
  "MessageId": "Base.1.0.InternalError",
  "Name": "System Event Log Entry",
  "Resolution": "ERoT Reset: Failed to unmount file system
towards ERoT reset.",
  "Resolved": false,
  "Severity": "Critical"
}
```

```
{
  "@odata.id":
"/redfish/v1/Systems/Bluefield/LogServices/EventLog/Entries/<I
...
  "MessageId": "Base.1.0.InternalError",
  "Name": "System Event Log Entry",
  "Resolution": "ERoT Reset: Failed to unmount file system. It
is still mounted as read-write (rw),
  "Resolved": false,
  "Severity": "Critical"
}
```

Resetting CEC and BMC Subsystems Using IPMI I2C Command over SMBus Channel Connected to PCIe Golden Finger

i Note

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

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

i Info

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

i Info

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

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

Resetting the Entire BlueField

This option typically involves a full power cycle of the host platform.

CEC Background Update Status

Info

This section is relevant only for BlueField-3.

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

Note

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

To check the status of the background update, run:

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

```

Info

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

Possible Error Codes

Info

This section is relevant only for BlueField-3.

Fault	Diagnosis and Possible Solution
<p>Connection to BMC breaks during firmware package transfer</p>	<ul style="list-style-type: none"> • Redfish task URI is not returned by the Redfish server • The Redfish server (if operational) is in <code>idle</code> state • After a reboot of BMC, or after restart/recovery of the Redfish server, the Redfish server is in <code>idle</code> state <p>A new firmware update can be attempted by the Redfish client.</p>
<p>Connection to BMC breaks during firmware update</p>	<ul style="list-style-type: none"> • Redfish task URI that was previously returned by the Redfish server is no longer accessible • The Redfish server (if operational) is in one of the following states: <ul style="list-style-type: none"> ◦ In <code>idle</code> state, if the firmware update has completed ◦ In <code>update</code> state, if the firmware update is still ongoing • After a BMC reboot, or after restart/recovery of the Redfish server, the Redfish server is in <code>idle</code> state <p>A new firmware update can be attempted by the Redfish client.</p>
<p>Two firmware update requests are initiated</p>	<p>The Redfish server blocks the second firmware update request and returns the following:</p> <ul style="list-style-type: none"> • HTTP code 400 "Bad Request" • Redfish message based on standard registry entry <code>UpdateInProgress</code> • A resolution is proposed: "Another update is in progress. Retry the update operation once it is complete." <p>Check the status of the ongoing firmware update by looking at the <code>TaskCollection</code> resource.</p>
<p>Redfish task hangs</p>	<ul style="list-style-type: none"> • Redfish task URI that was previously returned by the Redfish server is no longer accessible • PLDM-based firmware update progresses • After a reboot of BMC, or after restart/recovery of the Redfish server, the Redfish server is in <code>idle</code> state <p>A new firmware update can be attempted by the Redfish client.</p>

Fault	Diagnosis and Possible Solution
BMC-EROT communication failure during image transfer	<p>The Redfish task monitoring the firmware update indicates a failure:</p> <ul style="list-style-type: none"> • <code>TaskState</code> is set to <code>Exception</code> • <code>TaskStatus</code> is set to <code>Warning</code> • <code>Messages</code> array in the task includes an entry based on the standard registry <code>Update.1.0.0.TransferFailed</code> indicating the components that failed during image transfer <p>The Redfish client may retry the firmware update.</p>
Firmware update fails	<p>The Redfish task monitoring the firmware update indicates a failure:</p> <ul style="list-style-type: none"> • <code>TaskState</code> is set to <code>Exception</code> • <code>TaskStatus</code> is set to <code>Warning</code> • <code>Messages</code> array in the task includes an entry describing the error <p>The Redfish client may retry the firmware update.</p>
ERoT failure (not responding)	<p>The Redfish task monitoring the firmware update indicates a failure:</p> <ul style="list-style-type: none"> • <code>TaskState</code> is set to <code>Canceled</code> • <code>TaskStatus</code> is set to <code>Warning</code> • <code>Messages</code> array in the task includes an entry describing the error • The Redfish client reports the error <p>The Redfish client may retry the firmware update.</p>

Fault	Diagnosis and Possible Solution
Firmware image validation failure	<p>The Redfish task monitoring the firmware update indicates a failure:</p> <ul style="list-style-type: none"> • <code>TaskState</code> is set to <code>Exception</code> • <code>TaskStatus</code> is set to <code>Warning</code> • <code>Messages</code> array in the task includes an entry based on the standard registry <code>Update.1.0.0.VerificationFailed</code> to indicate the component for which verification failed • The Redfish client reports the error <p>The Redfish client might retry the firmware update.</p>
Power loss before activation command is sent	<ul style="list-style-type: none"> • The Redfish server is in <code>idle</code> state <p>A new firmware update can be attempted by the Redfish client.</p>
Firmware activation failure	<p>The Redfish task monitoring the firmware update indicates a failure:</p> <ul style="list-style-type: none"> • <code>TaskState</code> is set to <code>Exception</code> • <code>TaskStatus</code> is set to <code>Warning</code> • <code>Messages</code> array in the task includes an entry based on the standard registry <code>Update.1.0.ActivateFailed</code> <p>The Redfish client may retry the firmware update.</p>
Push to BMC firmware package greater than 200 MB	<ul style="list-style-type: none"> • No Redfish task is created • <code>Messages</code> array in the task includes an entry based on the standard registry <code>Base.1.15.0.PayloadTooLarge</code> and the <code>Resolution</code> "Firmware package size is greater than allowed size". Make sure the package size is less than the <code>UpdateService.MaxImageSizeBytes</code> property and retry the firmware update operation.

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 2026, NVIDIA. PDF Generated on 02/28/2026