Common Configurations

Table of contents

This section contains the following pages:

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/M
```

BIOS Secure Boot Configuration

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

Reading Secure Boot Status

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

Output example:

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

```
}
```

Setting Secure Boot State

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

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

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

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

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

second reboot is for the setting to take effect.

- 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/Secur
```

Output example:

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

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

> **Note**
>
> The following certificate is an example only and can not be used as is. `db` certificate must be signed by the public key certificate.

```
curl -k -u root:'<password>' -H 'Content-Type:
application/json' -X POST
https://<bmc_ip>/redfish/v1/Systems/Bluefield/SecureBoot/Secur
-d \
 '{"CertificateString": "-----BEGIN CERTIFICATE-----
\nMIIDbTCCAlWgAwIBAgIUO2MdJt2cTCGr0eO4PiBV5Uk0b/IwDQYJKoZIhvcN
----END CERTIFICATE-----","CertificateType": "PEM":
"5491316d-9694-4639-b72d-b8630ffa7dab"}'
```

Output example:

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

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

```
curl -k -u root:'<password>' -H 'Content-Type:
application/json' -X POST
https://<bmc_ip>/redfish/v1/Systems/Bluefield/SecureBoot/Secur
-d \
'{"SignatureString":
"80B4D96931BF0D02FD91A61E19D14F1DA452E66DB2408CA8604D411F92659
"UEFI","SignatureType": "EFI_CERT_SHA256_GUID": "28d5e212-
165b-4ca0-909b-c86b9cee0112"}'
```

Output example:

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

- To delete UEFI `db` certificate #1:

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

Output example:

```
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/2",
  "@odata.type": "#Task.v1_4_3.Task",
```

```
  "Id": "2",
  "TaskState": "Pending",
  "TaskStatus": "OK"
}
```

- To delete all UEFI `db` keys:

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

Output example:

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

Secure Boot Flow Example

The following is an example flow for resetting all `db` certificates using Redfish commands:

1. To reset all `db` keys:

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

```
https://<bmc_ip>/redfish/v1/Systems/Bluefield/SecureBoot/Secur
-d '{"ResetKeysType": "DeleteAllKeys"}'
```

Output example:

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

Tip

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

2. To read the status of task 12:

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

Output example:

```
{
  "@odata.id": "/redfish/v1/TaskService/Tasks/12",
  "@odata.type": "#Task.v1_4_3.Task",
  "Id": "12",
  "Messages": [],
```

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

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

3. Issue the following graceful reset command to BlueField :

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

Output example:

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

UEFI reads the pending secure boot tasks and executes them.

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

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

BIOS Configuration

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

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:

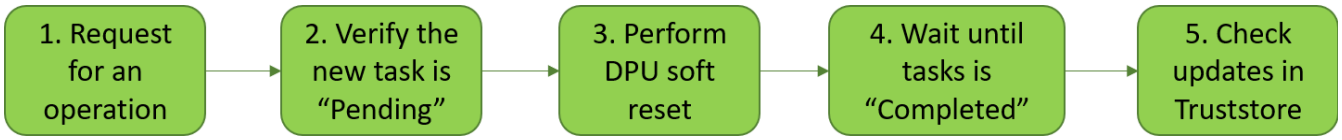
```
curl -k -u root:<password> -X GET
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Oem/Nvidia/Truststore/Certificates/<cert_num>
```

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": "Kg639IcpJtYMRzvh.nvidia",
        "Country": "US",
        "Organization": "NVIDIA",
        "OrganizationalUnit": "NBU",
        "State": "California"
    },
    "KeyUsage": [
        "CRLSigning"
    ],
```

```
  "Name" :  "TruststoreBios Certificate",
  "Subject" :  {
      "City" :  "Santa Clara",
      "CommonName" :  "Kg639IcpJtYMRzvh.nvidia",
      "Country" :  "US",
      "Organization" :  "NVIDIA",
      "OrganizationalUnit" :  "NBU",
      "State" :  "California"
  },
  "UefiSignatureOwner" :  "<UEFI_Owner>",
  "ValidNotAfter" :  "2043-01-01T00:00:00+00:00",
  "ValidNotBefore" :  "2023-01-01T00:00:00+00:00"
}
```

BIOS CA Certificates Collection Operations



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

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

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

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

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

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

      > **Note**
      >
      > The "\n" at the end are mandatory.

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

1.

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

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

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

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

2. Verify the new task is `Pending`:

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

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

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

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

4. Wait until task is `Completed`.

   The task details and status can be checked using the following `GET` request:

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

   The task status can be either:

   - Pending – Initial state.

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

     If the task remains in a "Pending" state after BlueField reset completes, please check the UEFI-BMC communication.

     If a communication failure occurs, consider either:

- Replacing the BMC certificate with one signed by a CA whose certificate was installed and resetting BlueField
- Removing all CA certificates from the UEFI menu

- Completed – Finished state.

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

- Exception – Failure state.

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

  In this case, verify that the certificate is valid.

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

BIOS Attributes

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

BIOS Debug Mode

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

- To enable the logs, run:

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

  Returns 01 if successful.

- To disable debug mode, run:

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

  Returns 00 if successful.

- To query the current applied mode, run:

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

  Returns:

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

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

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

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 NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA 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 2025. PDF Generated on 01/14/2025