



SoC Management Interface (RShim)

Table of contents

Installation and Upgrade

Configuration File

Host-side Interface Configuration

SoC Management Interface Driver Support for Multiple BlueFields

Virtual Ethernet Interface

Permanently Changing Arm-side MAC Address

Multi-board Management Example

SoC Management Interface Features and Functionality

RShim Misc Interface

BlueField Configuration File

RShim Ownership

Controlling SoC Management Interface from Using NC-SI Commands

Set Host PCIe RShim access to BlueField DPU CPU Response

Set Dynamic Host PCIe RShim access to BlueField DPU CPU (Command=0x12, Parameter=0x1B)

Get Host PCIe RShim access to BlueField DPU CPU (Command=0x13, Parameter=0x19)

Set Host PCIe RShim Access to BlueField DPU CPU (Command=0x12, Parameter=0x19)

Get Host PCIe RShim access to BlueField DPU CPU Response

Set Dynamic Host RShim Access to BlueField DPU CPU Response

The SoC management interface—formerly known as RShim—enables an external agent such as the host CPU or BMC to operate and monitor the NVIDIA® BlueField® platform (DPU or SuperNIC). This interface supports provisioning BlueField, resetting Arm cores, and retrieving logs.

Note

For instructions for Windows support, please refer to page "[Windows Support](#)".

Installation and Upgrade

For installation and upgrade instructions, see "[Updating Repo Package on Host Side](#)".

Configuration File

The configuration file for the SoC management interface is located at `/etc/rshim.conf` and includes the parameters shown in the table below:

Parameter	Default	Description
<code>BOOT_TIMEOUT</code>	150	Timeout value in seconds when pushing BFB while Arm side is not reading the boot stream. Timeout in seconds when pushing BFB while the Arm side is not reading the boot stream.
<code>DROP_MODE</code>	0	If set to 1, the RShim driver ignores all RShim writes and returns 0 for reads. Useful during <code>FW_RESET</code> or when bypassing the RShim PF to a VM.
<code>PCIE_RESET_DELAY</code>	10	Delay in seconds added after a chip reset and before pushing the boot stream when using RShim over PCIe.
<code>PCIE_INTR_POLL_INTERVAL</code>	10	Interrupt polling interval (in seconds) when using RShim over direct memory mapping.

Parameter	Default	Description
PCIE_HAS_VFIO	1	Set to 0 to disable RShim memory mapping via VFIO.
PCIE_HAS_UIO	1	Set to 0 to disable RShim memory mapping via UIO.

Note

RShim configuration is optional. The default parameters support out-of-the-box deployment scenarios, including setups with multiple BlueField devices on a single host.

To control which RShim index maps to which device, edit the configuration file as follows:

```
# Uncomment the 'rshim<N>' line to define the mapping.
#
# Format:
# rshim<N>    pci-device
rshim0        pci-0000:21:00.2
rshim1        pci-0000:81:00.2

# Ignored devices
# Uncomment the 'none' line to exclude devices.
#
#none         usb-1-1.4
#none         pci-lf-0000:84:00.0
```

Note

After making any changes to the configuration, restart the SoC management interface with:

```
systemctl restart rshim
```

Host-side Interface Configuration

On the host OS, BlueField registers a “DMA controller” used for management over PCIe. You can verify its presence with the following command:

```
# lspci -d 15b3: | grep 'SoC Management Interface'
```

Example output:

```
27:00.2 DMA controller: Mellanox Technologies MT42822 BlueField-2  
SoC Management Interface (rev 01)
```

A special driver—currently named **RShim**—must be installed and running on the host to expose the SoC management interface. This driver is automatically installed as part of the DOCA package.

For installation details, refer to the section **Install RShim on Host**.

When RShim is operating correctly, the host creates:

- A sysfs device node: `/dev/rshim0/*`
- A virtual Ethernet interface: `tmfifo_net0`

To check the driver status:

```
systemctl status rshim
```

Example output:

```
# systemctl status rshim
rshim.service - rshim driver for BlueField SoC
   Loaded: loaded (/lib/systemd/system/rshim.service; disabled;
 vendor preset: enabled)
   Active: active (running) since Tue 2022-05-31 14:57:07 IDT;
 1 day 1h ago
     Docs: man:rshim(8)
   Process: 90322 ExecStart=/usr/sbin/rshim $OPTIONS
 (code=exited, status=0/SUCCESS)
   Main PID: 90323 (rshim)
     Tasks: 11 (limit: 76853)
    Memory: 3.3M
    CGroup: /system.slice/rshim.service
           90323 /usr/sbin/rshim
```

Log entries confirm device initialization:

```
rshim[90323]: Probing pcie-0000:a3:00.2(vfio)
rshim[90323]: Create rshim pcie-0000:a3:00.2
rshim[90323]: rshim pcie-0000:a3:00.2 enable
rshim[90323]: rshim0 attached
```

Virtual Ethernet Interface

The RShim driver exposes a virtual Ethernet device, `tmfifo_net0`, which acts as a peer-to-peer tunnel between the host and BlueField OS. On the BlueField side, BFB images configure a static IP address of `192.168.100.2/30`. The host side must be configured manually.

To configure the host IP:

```
ip addr add dev tmfifo_net0 192.168.100.1/30
```

Note

For persistent configuration, see "Assign a static IP to `tmfifo_net0`" under "[Updating Repo Package on Host Side](#)".

At this point, to log into the BlueField OS from the host, run the following for example:

```
ssh ubuntu@192.168.100.2
```

SoC Management Interface Driver Support for Multiple BlueFields

If multiple BlueField devices are installed on the same host, each will have:

- A unique RShim interface at `/dev/rshim<N>`
- A corresponding virtual Ethernet device: `tmfifo_net<N>`

Note

`<N>` corresponds to the device index (starting from 0).

There are two ways to manage multiple `tmfifo_net<N>` interfaces:

- Bridge mode
 1. Add all `tmfifo_net<N>` interfaces to a bridge.
 2. Assign a single IP to the bridge.
 3. Each BlueField is configured with a unique IP in the bridge's subnet.
- Point-to-point mode:
 1. Assign a unique subnet IP to each `tmfifo_net<N>` interface.
 2. Each BlueField is assigned a corresponding IP in that subnet.

Each `tmfifo_net<N>` interface on the host must have a unique MAC address. To configure it manually:

```
ifconfig tmfifo_net0 192.168.100.1/24 hw ether 02:02:02:02:02:02
```

It can also be configured persistently using a custom udev rule.

Similarly, each BlueField-side `tmfifo_net` interface must be configured with:

- A unique MAC address
- A distinct IP (default is `192.168.100.2` on all devices)

These must be set manually or via BlueField [customization scripts](#) during installation.

Multi-board Management Example

This example deals with two BlueField devices installed on the same server (the process is similar for more devices). The example assumes that the RShim package has been installed on the host server.

Configuring Management Interface on Host

Note

This example is relevant for CentOS/RHEL operating systems only.

1. Create a `bf_tmfifo` interface under `/etc/sysconfig/network-scripts`. Run:

```
vim /etc/sysconfig/network-scripts/ifcfg-br_tmfifo
```

2. Inside `ifcfg-br_tmfifo`, insert the following content:

```
DEVICE="br_tmfifo"  
BOOTPROTO="static"  
IPADDR="192.168.100.1"  
NETMASK="255.255.255.0"  
ONBOOT="yes"  
TYPE="Bridge"
```

3. Create a configuration file for the first BlueField, `tmfifo_net0`. Run:

```
vim /etc/sysconfig/network-scripts/ifcfg-tmfifo_net0
```

4. Inside `ifcfg-tmfifo_net0`, insert the following content:

```
DEVICE=tmfifo_net0
BOOTPROTO=none
ONBOOT=yes
NM_CONTROLLED=no
BRIDGE=br_tmfifo
```

5. Create a configuration file for the second BlueField, `tmfifo_net1`. Run:

```
DEVICE=tmfifo_net1
BOOTPROTO=none
ONBOOT=yes
NM_CONTROLLED=no
BRIDGE=br_tmfifo
```

6. Create the rules for the `tmfifo_net` interfaces. Run:

```
vim /etc/udev/rules.d/91-tmfifo_net.rules
```

7. Restart the network for the changes to take effect. Run:

```
# /etc/init.d/network restart
Restarting network (via systemctl):          [ OK ]
```

Configuring BlueField Side

BlueField devices arrive with the following factory default configurations for `tmfifo_net0`.

Address	Value
MAC	00:1a:ca:ff:ff:01
IP	192.168.100.2

Therefore, if you are working with more than one BlueField, you must change the default MAC and IP addresses.

Updating RShim Network MAC Address

Note

This procedure is relevant for Ubuntu/Debian (`sudo` needed), and CentOS BFBs. The procedure only affects the `tmfifo_net0` on the Arm side.

1. Use a Linux console application (e.g. screen or minicom) to log into each BlueField. For example:

```
# sudo screen /dev/rshim<0|1>/console 115200
```

2. Create a configuration file for `tmfifo_net0` MAC address. Run:

```
# sudo vi /etc/bf.cfg
```

3. Inside `bf.cfg`, insert the new MAC:

```
NET_RSHIM_MAC=00:1a:ca:ff:ff:03
```

4. Apply the new MAC address. Run:

```
sudo bfcfg
```

5. Repeat this procedure for the second BlueField (using a different MAC address).

Info

Arm must be rebooted for this configuration to take effect. It is recommended to update the IP address before you do that to avoid unnecessary reboots.

Note

For comprehensive list of the supported parameters to customize `bf.cfg` during BFB installation, refer to section "[bf.cfg Parameters](#)".

Updating IP Address

For Ubuntu:

1. Access the file `50-cloud-init.yaml` and modify the `tmfifonet0` IP address:

```
sudo vim /etc/netplan/50-cloud-init.yaml

        tmfifo_net0:
            addresses:
                - 192.168.100.2/30      ==>>>
192.168.100.3/30
```

2. Reboot the Arm. Run:

```
sudo reboot
```

3. Repeat this procedure for the second BlueField (using a different IP address).

Info

Arm must be rebooted for this configuration to take effect. It is recommended to update the MAC address before you do that to avoid unnecessary reboots.

For CentOS:

1. Access the file `ifcfg-tmfifo_net0`. Run:

```
# vim /etc/sysconfig/network-scripts/ifcfg-tmfifo_net0
```

2. Modify the value for `IPADDR`:

```
IPADDR=192.168.100.3
```

3. Reboot the Arm. Run:

```
reboot
```

Or perform `netplan apply`.

4. Repeat this procedure for the second BlueField (using a different IP address).

Info

Arm must be rebooted for this configuration to take effect. It is recommended to update the MAC address before you do that to avoid unnecessary reboots.

Permanently Changing Arm-side MAC Address

Note

It is assumed that the commands in this section are executed with root (or `sudo`) permission.

The default MAC address is `00:1a:ca:ff:ff:01`. It can be changed using `ifconfig` or by updating the UEFI variable as follows:

1. Log into Linux from the Arm console.
2. Run:

```
$ "ls /sys/firmware/efi/efivars".
```

3. If not mounted, run:

```
$ mount -t efivarfs none /sys/firmware/efi/efivars
$ chattr -i /sys/firmware/efi/efivars/RshimMacAddr-8be4df61-93ca-11d2-aa0d-00e098032b8c
$ printf "\x07\x00\x00\x00\x00\x1a\xca\xff\xff\x03" > \
  /sys/firmware/efi/efivars/RshimMacAddr-8be4df61-93ca-11d2-aa0d-00e098032b8c
```

The `printf` command sets the MAC address to `00:1a:ca:ff:ff:03` (the last six bytes of the `printf` value). Either reboot the device or reload the `tmfif0` driver for the change to take effect.

The MAC address can also be updated from the server host side while the Arm-side Linux is running:

1. Enable the configuration. Run:

```
# echo "DISPLAY_LEVEL 1" > /dev/rshim0/misc
```

2. Display the current setting. Run:

```
# cat /dev/rshim0/misc
DISPLAY_LEVEL 1 (0:basic, 1:advanced, 2:log)
BOOT_MODE     1 (0:rshim, 1:emmc, 2:emmc-boot-swap)
BOOT_TIMEOUT  300 (seconds)
DROP_MODE     0 (0:normal, 1:drop)
SW_RESET      0 (1: reset)
DEV_NAME      pcie-0000:04:00.2
DEV_INFO      BlueField-2(Rev 1)
PEER_MAC      00:1a:ca:ff:ff:01 (rw)
PXE_ID        0x00000000 (rw)
VLAN_ID       0 0 (rw)
```

3. Modify the MAC address. Run:

```
$ echo "PEER_MAC xx:xx:xx:xx:xx:xx" > /dev/rshim0/misc
```

Info

For more information and an example of the script that covers the installation and configuration of multiple BlueField devices, refer to section "[Installing Full DOCA Image on Multiple BlueField Platforms](#)" of the *NVIDIA DOCA Installation Guide*.

SoC Management Interface Features and Functionality

	Function	Command	Comments
1	Push BFB	<pre>bfb-install -r rshim<N> -b <bfb> [-c bf.cfg]</pre>	Using <code>bfb</code> command more details, refer to BlueField Deployment
2	Open console	<pre>screen /dev/rshim<N>/console 115200 minicom -D /dev/rshim<N>/console</pre>	The <code>N</code> in the number of devices in Use Linux minicom application BlueField
3	Configure a virtual network interface	<pre>ip addr add dev tmfifo_net<N> 192.168.100.1/30</pre>	The <code>N</code> in the number of devices in to section Management Driver Setup BlueField information. The default BlueField IP is 192.168. The IP is (192.168. example
4	Log into the DPU	<pre>ssh -6 user@fe80::21a:caff:feff:ff01%tmfifo_net<N></pre>	The <code>N</code> in the number of devices in setup. Refer to Management Driver Setup BlueField information.
5	PXE boot over RShim	N/A	Please refer to "Deploying Software over PXE" for more details.

	Function	Command	Comment
6	Check Arm/Rshim status	<pre>cat /dev/rshim<N>/misc</pre>	Example

	Function	Command	Comment
			<pre> \$ su /dev DISP (0:b 1:ad BF_M NIC BOOT (0:r 2:eml BOOT 300 (USB_ (sec DROP (0:n SW_R (1: DEV_ pcie DEV_ Blue OPN_ 9009 UP_T 59537 SECU (0:n FORC (1: comm ---- ----- </pre>

	Function	Command	Comment
			<pre> --- Mess ---- ---- --- INF up INF is r </pre> <p>For detail individual section "Interface"</p>
7	Configure Arm/RShim	<pre> echo "<command string>" > /dev/rshim<N>/misc </pre>	<p>See section Interface software</p> <pre> echo > /dev </pre>
8	Expose log messages	N/A	For more refer to s

	Function	Command	Comment
9	Update RShim PCIe device name dynamically	<pre>echo DEV_NAME > /dev/rshim0/misc</pre>	<p>This command updates the RShim device name according to the PCIe function PF2 has just installed. PF2 has just one of the PCIe functions from <code>xx:xx:xx:xx:xx:xx</code> to <code>xx:xx:xx:xx:xx:xx</code>. RShim device is updated. Restarting the service—disrupting all BlueField commands—only the device name, minimizing interruption.</p>

RShim Misc Interface

Field	Description	Values	Writable?	Shown in Display Level
DISPLAY_LEVEL	Verbosity of the RShim misc output	<ul style="list-style-type: none"> • 0 – Basic • 1 – Advanced (level 0 info plus advanced configuration) • 2 – Log (level 0 info plus RShim log) 	Yes	All
BF_MODE	Check whether BlueField DPU is booting in DPU mode or NIC mode	<ul style="list-style-type: none"> • Unknown – The mode value has not been set by the BIOS (ATF). This value can be seen on older versions of BlueField-3 BIOS firmware. • DPU mode – BlueField is running in DPU mode • NIC mode – BlueField is running in NIC Mode • Reserved – Unknown mode value reserved for future use 	No	All

Field	Description	Values	Writable?	Shown in Display Level
BOOT_MODE	Boot location of BlueField Arm software	<ul style="list-style-type: none"> 0 – RShim 1 – emmc 2 – emmc-boot-swap 	Yes	All
BOOT_TIMEOUT	Defines the timeout (in seconds) for writing a BFB file to the RShim boot device file. While the BFB file is being written, the BlueField firmware is expected to continuously consume the data. If data consumption stops and the timeout is reached, the write operation to the boot device file will fail with an error.	300 seconds (default)	Yes	All
USB_TIMEOUT	Specifies the timeout (in seconds) for writing to the boot device file when RShim is using a USB backend. USB transfers—especially on BlueField-2—can be slower than PCIe. If data cannot be written within this timeout period, the RShim driver aborts the operation.	40 seconds (default)	Yes	All

Field	Description	Values	Writable?	Shown in Display Level
DROP_MODE	Indicates whether the RShim driver is operating in "drop" mode. In this mode, the driver relinquishes control of the RShim hardware and disables most RShim functionalities. This is typically used in scenarios where another entity temporarily assumes ownership of the RShim interface.	<p>0 – drop mode disabled</p> <p>1 – drop mode enabled</p>	Yes	All
SW_RESET	This is a write-only field that provides a way to reset DPU Arm. Example: <pre>echo "SW_RESET 1" > /dev/rshim0/misc</pre>	Always shown as 0.	Yes Accepted values: 1	All
DEV_NAME	RShim driver/backend name	<p>Example:</p> <p>pcie-0000:b1:00.2 on PCIe host and usb-2.1</p>	No	All
DEV_INFO	BlueField version and revision	<p>Example:</p> <p>BlueField-3(Rev 1)</p>	No	All
OPN_STR	BlueField OPN string	<p>Example:</p> <p>9009D3B600CVAA</p>	No	All
UP_TIME	Time since BlueField boots up	<p>Example:</p> <p>597068(s)</p>	No	All

Field	Description	Values	Writable?	Shown in Display Level
<code>SECURE_NIC_MODE</code>	Whether RShim is in secure NIC mode (also called "locked" mode) to prevent host RShim access	<input type="checkbox"/> – no <input type="checkbox"/> – yes	No	All
<code>FORCE_CMD</code>	Indicates whether a FORCE command is pending. This command is used to request ownership transfer of the RShim interface from the other backend—PCIe (host) or USB (BMC).	<input type="checkbox"/> – Force command pending <input type="checkbox"/> – No FORCE command is pending. This triggers an RShim FORCE command to initiate ownership transfer.	Yes	All
(RShim log messages)			No	2

BlueField Configuration File

The `bf.cfg` file contains configuration that can be pushed to customize the installation of the BFB. See "[Customizing BlueField Software Deployment](#)" for more information.

RShim Ownership

The RShim interface may be owned by the BlueField BMC or the host (Windows or Linux). In situations where users do not have access to the host, they would want to transfer RShim ownership to the BMC.

Assuming that `/dev/rshim0` is the BlueField requesting ownership over the RShim interface, ownership may be transferred to the BlueField BMC by running the following command from the BMC console:

1. Confirm RShim is not attached to BMC:

```
root@dpu-bmc:~# systemctl status rshim
```

This show messages like `another backend already attached` or `rshim0 entering drop mode`.

Info

If RShim is already attached or RShim is not in drop mode, the ownership transfer command (`-F`) will fail.

2. Create a directory for the RShim systemd override file if it does not already exist:

```
root@dpu-bmc:~# mkdir -p /etc/systemd/system/rshim.service.d
```

3. Create the override file:

```
root@dpu-bmc:~# cat >
/etc/systemd/system/rshim.service.d/override.conf
[Service]
Environment="OPTIONS=-F"
```

4. Press Ctrl-D to save the file.
5. Reload the systemd manager configuration and restart RShim service.

```
root@dpu-bmc:~# systemctl daemon-reload
root@dpu-bmc:~# systemctl restart rshim
root@dpu-bmc:~# systemctl status rshim
... ..
Dec 10 20:58:57 dpu-bmc rshim[20561]: rshim0 received
ownership transfer ack
Dec 10 20:59:00 dpu-bmc rshim[20561]: rshim0 regained
ownership successfully
```

RShim ownership is transferred to BMC.

6. Clean up the systemd override file:

```
root@dpu-bmc:~# rm
/etc/systemd/system/rshim.service.d/override.conf
root@dpu-bmc:~# systemctl daemon-reload
```

Controlling SoC Management Interface from Using NC-SI Commands

Controlling SoC management interface could also be done from the platform BMC using NC-SI OEM command over I²C.

RShim interface could be enabled, disabled, or locked:

- Enabled – RShim over PCIe is supported and ready to use
- Locked – RShim over PCIe is supported, but cannot currently be used. It may be re-activated using the enable command, without the need to perform reset
- Disabled – RShim over PCIe is not supported. You must re-enable it and power cycle the host.

Set Host PCIe RShim Access to BlueField DPU CPU (Command=0x12, Parameter=0x19)

This package command allows a trusted platform BMC to configure the non-volatile enablement state of external Host PCIe RShim access to the BlueField DPU's embedded CPU.

Note

A BlueField DPU reboot is required for this setting to take effect.

Set Host PCIe RShim Access to BlueField DPU CPU Format

Bytes/Bits	31:24	23:16	15:8	7:0
0:15	NC-SI Header (OEM Command)			
16:19	NVIDIA Manufacture ID (IANA) = 0x8119			
20:23	Command rev=0x00	MLNX Cmd ID=0x12	Parameter=0x19	Reserved
24:27	Reserved			Host_Access_State
28:31	Checksum 31:0			

Set Host PCIe RShim Access to BlueField DPU CPU Command Parameters

Field	Bytes	Offset in NC-SI Command	Description
Host_Access_State	1	27	Embedded CPU OS state <ul style="list-style-type: none"> • 0 - Disabled • 1 - Enabled • 2 - Locked (PF Enabled, access is disabled) • Other - reserved

Set Host PCIe RShim access to BlueField DPU CPU Response

The BlueField DPU always receives and responds to this command when the Package ID matches.

If the command is issued by an untrusted platform BMC, it fails with reason code `0x7FFF` (Unsupported command).

Set Host RShim Access to BlueField DPU CPU Response Format

Bytes/Bits	31:24	23:16	15:8	7:0
0:15	NC-SI Header (OEM Command)			
16:19	Response Code		Reason Code	
20:23	NVIDIA Manufacture ID (IANA) = 0x8119			
24:27	Command rev=0x00	MLNX Cmd ID=0x12	Parameter=0x19	Reserved
28:31	Reserved			Host_Access_State
32:35	Checksum 31:0			

Set Dynamic Host PCIe RShim access to BlueField DPU CPU (Command=0x12, Parameter=0x1B)

This package command allows a trusted platform BMC to configure the runtime (volatile) enablement state of Host PCIe RShim access to the BlueField DPU CPU—without requiring a DPU reset.

***i* Note**

This command is only valid when the non-volatile setting is `Enabled` or `Locked` (see section "[Set Host PCIe RShim Access](#)").

The setting applied by this command is volatile and will be overridden upon reset by the non-volatile configuration.

Set Dynamic Host PCIe RShim Access to BlueField DPU CPU Format

Bytes/Bits	31:24	23:16	15:8	7:0
0:15	NC-SI Header (OEM Command)			
16:19	NVIDIA Manufacture ID (IANA) = 0x8119			
20:23	Command rev=0x00	MLNX Cmd ID=0x12	Parameter=0x1B	Reserved
24:27	Reserved			Host_RT_Access_State
28:31	Checksum 31:0			

Set Dynamic Host PCIe RShim Access to BlueField DPU CPU Command Parameters

Field	Bytes	Offset in NC-SI Command	Description
Host_RT_Access_State	1	27	Embedded CPU OS state <ul style="list-style-type: none"> • 0 - Enabled • 1 - Locked • Other - reserved

Set Dynamic Host RShim Access to BlueField DPU CPU Response

The BlueField DPU processes this command only when:

- The Package ID matches
- The non-volatile setting is `Enabled` or `Locked`

If the non-volatile setting is `Disabled`, the command fails with reason code `0x0002` (Command unavailable).

If issued by an untrusted platform BMC, the command fails with reason code `0x7FFF` (Unsupported command).

Note

This command supports toggling between **Enabled** and **Locked** states without requiring a DPU reboot.

Set Dynamic Host RShim Access to BlueField DPU CPU Response Format

Bytes/Bits	31:24	23:16	15:8	7:0
0:15	NC-SI Header (OEM Command)			
16:19	Response Code		Reason Code	
20:23	NVIDIA Manufacture ID (IANA) = 0x8119			
24:27	Command rev=0x00	MLNX Cmd ID=0x12	Parameter=0x1B	Reserved
28:31	Reserved			Host_RT_Access_State
32:35	Checksum 31:0			

Get Host PCIe RShim access to BlueField DPU CPU (Command=0x13, Parameter=0x19)

This package command allows the platform BMC to query the BlueField DPU for the current enablement state of Host PCIe RShim access to the embedded CPU.

Get Host PCIe RShim Access to BlueField DPU CPU Format

Bytes/Bits	31:24	23:16	15:8	7:0
0:15	NC-SI Header (OEM Command)			
16:19	NVIDIA Manufacture ID (IANA) = 0x8119			
20:23	Command rev=0x00	MLNX Cmd ID=0x13	Parameter=0x19	Reserved
24:27	Checksum 31:0			

Get Host PCIe RShim access to BlueField DPU CPU Response

The BlueField DPU always receives and responds to this command when the Package ID matches.

Get Host PCIe RShim Access to BlueField DPU CPU Response

Field	Size	Offset in NC-SI Command	Description
Host_Access_State	1 byte	31	Embedded CPU OS state <ul style="list-style-type: none"> • 0 - Disabled • 1 - Enabled • Other - reserved

Get Host PCIe RShim Access to BlueField DPU CPU Response Format

Bytes/Bits	31:24	23:16	15:8	7:0
0:15	NC-SI Header (OEM Command)			
16:19	Response Code		Reason Code	
20:23	NVIDIA Manufacture ID (IANA) = 0x81 19			
24:27	Command rev=0x00	MLNX Cmd ID=0x13	Parameter=0x19	Reserved
28:31	Reserved			Host_Access_State
32:35	Checksum 31:0			

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