SoC Management Interface

Table of contents

The SoC management interface, formerly known as RShim, allows an external agent such as the host CPU or BMC to operate the NVIDIA® BlueField® networking platform (DPU or SuperNIC) and monitor its operational state. This interface allows provisioning of BlueField, resetting Arm cores, and obtaining logs.

> **Note**
>
> For instructions for Windows support, please refer to page "Windows Support".

Installation and Upgrade

Please refer to section 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 listed in the table below.

Parameter	Default	Description
BOOT_TIMEOUT	150	Timeout value in seconds when pushing BFB while Arm side is not reading the boot stream.
DROP_MODE	0	Once set to 1, the RShim driver ignores all RShim writes and returns 0 for RShim read. This is used in cases such as during FW_RESET or bypassing the RShim PF to VM.
PCIE_RESET_DELAY	10	Delay in seconds for RShim over PCIe, which is added after chip reset and before pushing the boot stream.
PCIE_INTR_POLL_INTERVAL	10	Interrupt polling interval in seconds when running RShim over direct memory mapping.
PCIE_HAS_VFIO	1	Setting this parameter to 0 disallows RShim memory mapping via

Parameter	Default	Description
		VFIO.
PCIE_HAS_UIO	1	Setting this parameter to 0 disallows RShim memory mapping via UIO.

Note

Configuring RShim is optional. The default parameters are designed to support out-of-box deployment scenarios including multiple BlueField device s on a single host.

Users may control which RShim index maps to which device by following this procedure:

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

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

Note

If any of these configurations are changed, then the SoC management interface must be restarted by running:

```
systemctl restart rshim
```

Host-side Interface Configuration

BlueField registers on the host OS a "DMA controller" for BlueField management over PCIe. This can be verified by running the following:

```
#  lspci -d 15b3: | grep 'SoC Management Interface'
27:00.2 DMA controller: Mellanox Technologies MT42822 BlueField-2 SoC Management Interface (rev 01)
```

A special SoC management driver must be installed and run on the host OS to expose the various BlueField management interfaces to the OS. Currently, this driver is named RShim and is automatically installed as part of the DOCA installation. Refer to section "Install RShim on Host" for information on how to obtain and install the host-side SoC management interface driver .

When the SoC management interface driver runs properly on the host side, a sysfs device, /dev/rshim0/*, and a virtual Ethernet interface, tmfifo_net0, become available. The following is an example for querying the status of the SoC management interface driver on the host side:

```
# 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
```

```
May 31 14:57:07 ...  systemd[1]: Starting rshim driver for BlueField SoC...
May 31 14:57:07  ... systemd[1]: Started rshim driver for BlueField SoC.
May 31 14:57:07  ... rshim[90323]: Probing pcie-0000:a3:00.2(vfio)
May 31 14:57:07  ... rshim[90323]: Create rshim pcie-0000:a3:00.2
May 31 14:57:07  ... rshim[90323]: rshim pcie-0000:a3:00.2 enable
May 31 14:57:08  ... rshim[90323]: rshim0 attached
```

If the SoC management interface driver device does not appear, refer to section "RShim Troubleshooting and How-Tos".

Virtual Ethernet Interface

On the host, the SoC management interface driver exposes a virtual Ethernet device called tmfifo_net0. This virtual Ethernet can be thought of as a peer-to-peer tunnel connection between the host and the BlueField OS. The BlueField OS also configures a similar device. The BlueField OS's BFB images are customized to configure the BlueField side of this connection with a preset IP of 192.168.100.2/30. It is up to the user to configure the host side of this connection. Configuration procedures vary for different OSs.

The following example configures the host side of tmfifo_net0 with a static IP and enables IPv4-based communication to the BlueField OS:

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

Note

For instructions on persistent IP configuration of the tmfifo_net0 interface, refer to step "Assign a static IP to tmfifo_net0" under "Updating Repo Package on Host Side".

Logging in from the host to the BlueField OS is now possible over the virtual Ethernet. For example:

```
ssh ubuntu@192.168.100.2
```

SoC Management Interface Driver Support for Multiple BlueFields

Multiple BlueField device s may connect to the same host machine. When the SoC management interface driver is loaded and operating correctly, each BlueField device is expected to have its own device directory on sysfs, /dev/rshim<N>, and a virtual Ethernet device, tmfifo_net<N>.

> **Note**
>
> <N> correlates to the number of BlueField devices used where the SoC management interfaces of the first BlueField is 0, incrementing by 1 for each added device.

The following are some guidelines on how to set up the SoC management virtual Ethernet interfaces properly if multiple BlueField device s are installed in the host system.

There are two methods to manage multiple tmfifo_net interfaces on a Linux platform:

- Using a bridge, with all tmfifo_net<N> interfaces on the bridge – the bridge device bares a single IP address on the host while each BlueField has unique IP in the same subnet as the bridge
- Directly over the individual tmfifo_net<N> – each interface has a unique subnet IP and each BlueField has a corresponding IP per subnet

Whichever method is selected, the host-side tmfifo_net interfaces should have different MAC addresses, which can be:

- Configured using ifconfig. For example:

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

- Or saved in configuration via the /udev/rules as can be seen later in this section.

In addition, each Arm-side tmfifo_net interface must have a unique MAC and IP address configuration, as BlueField OS comes uniformly pre-configured with a generic MAC, and 192.168.100.2. The latter must be configured in each BlueField manually or by BlueField customization scripts during BlueField OS 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 tmfifo_net0 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 tmfifo 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	`bfb-install –r rshim<N> -b <bfb> [-c bf.cfg]`	Using bf.cfg in the command is optional. For more details about bf.cfg, refer to "Customizing BlueField Software Deployment Using bf.cfg".
2	Open console	`screen /dev/rshim<N>/console 115200` `minicom -D /dev/rshim<N>/console`	The N index depends on the number of BlueField devices in your setup. Use Linux's screen or minicom console applications to access the BlueField console.
3	Configure a virtual network interface	`ip addr add dev tmfifo_net<N> 192.168.100.1/30`	The N index depends on the number of BlueField devices in your setup. Refer to section "SoC Management Interface Driver Support for Multiple BlueFields" for more information. The default IP address for the BlueField is 192.168.100.2/30. The IP used in the command (192.168.100.1/30) is for example purposes only.

	Function	Command	Comments
4	Log into the DPU	`ssh -6 user@fe80::21a:caff:feff:ff01%tmfifo_net<N>`	The N index depends on the number of DPUs in your setup. Refer to section "SoC Management Interface Driver Support for Multiple BlueFields" for more information.
5	PXE boot over RShim	N/A	Please refer to section "Deploying BlueField Software Using BFB with PXE" for more information.
6	Issue Arm software reset	`echo "SW_RESERT 1" > /dev/rshim<N>/misc`	
7	Expose log messages	N/A	For more information, please refer to section "Logging".

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 Using bf.cfg" 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:

```
bf-bmc# echo "FORCE_CMD 1" > /dev/rshim0/misc
```

Tip

This method requires the RShim driver binary to be run with -F or --force option, or have the FORCE_MODE set to 1 in rshim.conf. Otherwise, /dev/rshim<N>/ would not be created if the driver is detached from RShim.

To set ownership priority to the host or BMC, users may forcibly do so using either of the following options:

- The command line option -F or --force
- Setting FORCE_MODE 1 in the configuration file rshim.conf