



# **Intelligent Platform Management Interface**

# Table of contents

## BlueField BMC IPMI Commands

---

## BlueField IPMI Commands

---

I2C Addresses for BMC-initiated Requests

---

Changing I2C Addresses

---

I2C Addresses for BlueField-initiated Requests

---

## Disabling IPMI from BlueField Arm to BMC

---

Disabling IPMI Using UEFI Menu

---

Disabling IPMI Using Redfish

---

## External Host IPMI Commands

---

## Loading and Using IPMI on BlueField Running CentOS

---

IPMB requests can be initiated in 2 directions:

- BlueField BMC-to-BlueField
- BlueField-to-BlueField BMC

### **Note**

The NVIDIA® BlueField® networking platform's (DPU or SuperNIC) `ipmb_dev_int` driver is registered at the 7-bit I<sup>2</sup>C address 0x30 by default. The I<sup>2</sup>C address of the BlueField can be changed in the file `/usr/bin/set_emu_param.sh`.

- BlueField Controller cards provide connection from the host server BMC to BlueField Arm I<sup>2</sup>C bus
- BlueField devices provide connection from the host server BMC to the BlueField NC-SI port
- BlueField Reference Platforms provide connection from its on-board BMC to BlueField Arm I<sup>2</sup>C bus

## **BlueField BMC IPMI Commands**

The BlueField BMC is able to retrieve data from BlueField software over its Intelligent Platform Management Bus (IPMB).

For more details, please refer to the following sections of the NVIDIA BlueField BMC Software user manual under [BMC SW](#) tab.

- Appendix - Generic IPMI Commands
- Appendix - NVIDIA OEM IPMI Commands

## **BlueField IPMI Commands**

The BlueField is able to retrieve data from the BlueField BMC over IPMB.

Issue a command with the following format from the BlueField to retrieve information from the BMC:

```
$ ipmitool <ipmitool command>
```

The BlueField may request information about itself using the following command format:

```
$ ipmitool -U ADMIN -P ADMIN -p 9001 -H localhost <ipmitool command>
```

### **i Note**

The `ipmb_host` driver allows the BlueField to send requests to the BMC. Once `set_emu_param.service` is started, it will try to load the `ipmb_host` drivers. If the BMC is down or not responsive when BlueField tries to load the `ipmb_host` driver, the latter will not load successfully. In that case, make sure the BMC is up and operational, and run the following from BlueField's console:

```
echo 0x1011 > /sys/bus/i2c/devices/i2c-2/delete_device
rmmod ipmb_host
```

The `set_emu_param.service` script will try to load the driver again.

## **I2C Addresses for BMC-initiated Requests**

Device	I <sup>2</sup> C Address
BlueField <code>ipmb_dev_int</code>	0x30
BMC <code>ipmb_host</code>	0x20

## I2C Addresses for BlueField-initiated Requests

Device	I <sup>2</sup> C Address
BlueField <code>ipmb_host</code>	0x11
BMC <code>ipmb_dev_int</code>	0x10

## Changing I2C Addresses

To use a different BlueField or BMC I<sup>2</sup>C address, you must make changes to the following files' variables.

Filename Path	Parameter Change
<pre data-bbox="164 590 651 632">/usr/bin/set_emu_param.sh</pre>	<p data-bbox="678 233 1421 317">The <code>ipmb_dev_int</code> and <code>ipmb_host</code> drivers are registered at the following I<sup>2</sup>C addresses:</p> <ul data-bbox="722 352 1458 531" style="list-style-type: none"> <li data-bbox="722 352 1458 443">• <code>IPMB_DEV_INT_ADD=&lt;BlueField I2C Address 1&gt;</code></li> <li data-bbox="722 443 1458 531">• <code>IPMB_HOST_ADD=&lt;BlueField I2C Address 2&gt;</code></li> </ul> <p data-bbox="678 569 1463 642">These addresses must be different from one another. Otherwise, one of the drives will fail to register.</p> <p data-bbox="678 653 1154 688">To change the BMC I<sup>2</sup>C address:</p> <pre data-bbox="737 751 1328 936">IPMB_HOST_CLIENTADDR=&lt;BMC I2C Address&gt; &lt;I2C Address&gt; must be equal to: 0x1000+&lt;7-bit I2C address&gt;</pre>

## Disabling IPMI from BlueField Arm to BMC

The BlueField SoC features two I2C channels connecting BlueField Arm and the BMC:

- I2C-1 sends IPMI commands from BlueField Arm to the BMC
- I2C-5 sends IPMI commands from the BMC to BlueField Arm

In cases where the BlueField Arm is not trusted, it may be desirable to block IPMI commands from the BlueField Arm to the BMC as it could grant a malicious actor root-level access to the BMC. This can be done via Redfish or the UEFI menu.

## Disabling IPMI Using Redfish

1. Disable I2C-1:

```
curl -k -u root:'bmc_password' -H 'content-type: application/json' -d '{"Attributes":
{"DisableI2c1": true}}' -X PATCH
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Bios/Settings
```

2. Make sure the configuration is saved in the BMC's data base:

```
# curl -k -u root:'bmc_password' -H 'content-type: application/json' X GET
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Bios/Settings{
  "@odata.id": "/redfish/v1/Systems/Bluefield/Bios/Settings",
  "@odata.type": "#Bios.v1_2_0.Bios",
  "Attributes": {
    "DisableI2c1": true
  },
  "Description": "BIOS Settings",
  "Id": "BIOS_Settings",
  "Name": "BIOS Configuration"
}
```

3. Perform warm reset (e.g., `SW_RESET`) and wait for Redfish to run.
4. Perform warm reset (e.g., `SW_RESET`) and wait for Linux to boot.
5. Verify that `i2c-1` has successfully been disabled:

```
root@localhost:~# ls /sys/bus/i2c/devices/
```

Expected output:

```
i2c-5
```

i2c-1 device should not appear.

## Disabling IPMI Using UEFI Menu

1. Select Device Manager:

```
BlueField-3 SmartNIC Main Card
Mellanox BlueField-3 [A1] A78(D42) r0p1          2.00 GHz
4.9.0.0                                           15840 MB RAM

  Select Language          <Standard English>      This is the option
  > Device Manager        one adjusts to change
  > Boot Manager          the language for the
  > Boot Maintenance Manager current system

  Continue
  Reset

^v=Move Highlight      <Enter>=Select Entry
```

2. Select System Configuration:





It is possible for the external host to retrieve data from the BlueField via the IPMI LAN interface (either OOB or ConnectX).

To do that:

1. Set the network interface address properly in `progconf`. For example, if the OOB IP address is 192.168.101.2, edit the `OOB_IP` variable in the `/etc/ipmi/progconf` file as follows:

```
root@localhost:~# cat /etc/ipmi/progconf
SUPPORT_IPMB="NONE"
LOOP_PERIOD=3
BF_FAMILY=$(/usr/bin/bffamily | tr -d '[:space:]')
OOB_IP="192.168.101.2"
```

2. Then reboot or restart the IPMI service as follows:

```
systemctl restart mlx_ipmid
```

3. To get information from the BlueField, issue commands from the external host in the following format:

```
ipmitool -I lanplus -H 192.168.101.2 -U ADMIN -P ADMIN <ipmitool
command>
```

## Loading and Using IPMI on BlueField Running CentOS

1. Load the BlueField CentOS image:

## **(i) Note**

The following steps are performed from the BlueField CentOS prompt. The BlueField is running CentOS 7.6 with kernel 5.4. The CentOS installation was done using the CentOS everything ISO image.

The following drivers need to be loaded on the BlueField running CentOS:

- `jc42.ko`
- `ee1004.ko`
- `at24.ko`
- `eeeprom.ko`
- `i2c-dev.ko`

Example of loading `ee1004.ko`, `at24.ko`, and `eeeprom.ko`:

```
modprobe ee1004
modprobe at24
modprobe eeeprom
```

## **(i) Info**

The `i2c-dev` module is built into the kernel 5.4.60 on CentOS 7.6.

2. (Optional) Update the `i2c-mlx` driver if the installed version is older than `i2c-mlx-1.0-0.gab579c6.src.rpm`.

1. Re-compile `i2c-mlx`. Run:

```
$ yum remove -y kmod-i2c-mlx
$ modprobe -rv i2c-mlx
```

2. Transfer the `i2c-mlx` RPM from the BlueField software tarball under `distro/SRPM` onto the Arm. Run:

```
$ rpmbuild --rebuild /root/i2c-mlx-1.0-0.g422740c.src.rpm
$ yum install -y /root/rpmbuild/RPMS/aarch64/i2c-mlx-1.0-0.g422740c_5.4.17_mlnx.9.ga0bea68.aarch64.rpm
$ ls -l /lib/modules/$(uname -r)/extra/i2c-mlx/i2c-mlx.ko
```

3. Load `i2c-mlx`. Run:

```
$ modprobe i2c-mlx
```

3. Install the following packages:

```
$ yum install ipmitool lm_sensors
```

If the above operation fails for `ipmitool`, run the following to install it:

```
wget
http://sourceforge.net/projects/ipmitool/files/ipmitool/1.8.18
1.8.18.tar.gz
tar -xvzf ipmitool-1.8.18.tar.gz
cd ipmitool-1.8.18
./bootstrap
./configure
make
make install DESTDIR=/tmp/package-ipmitool
```

4. The `i2c-tools` package is also required, but the version contained in the CentOS Yum repository is old and does not work with BlueField. Therefore, please download `i2c-tools` version 4.1, and then build and install it.

```
# Build i2c-tools from a newer source
wget http://mirrors.edge.kernel.org/pub/software/utils/i2c-
tools/i2c-tools-4.1.tar.gz
tar -xvzf i2c-tools-4.1.tar.gz
cd i2c-tools-4.1
make
make install PREFIX=/usr

# create a link to the libraries
ln -sf /usr/lib/libi2c.so.0.1.1 /lib64/libi2c.so
ln -sf /usr/lib/libi2c.so.0.1.1 /lib64/libi2c.so.0
```

5. Generate an RPM binary from the BlueField's `mlx-OpenIPMI-2.0.25` source RPM.

The following packages might be needed to build the binary RPM depending on which version of CentOS you are using.

```
$ yum install libtool rpm-devel rpmdevtools rpmlint wget
ncurses-devel automake
$ rpmbuild --rebuild mlx-OpenIPMI-2.0.25-0.g581ebbb.src.rpm
```

**(i) Note**

You may obtain this rpm file by means of scp from the server host's Bluefield Distribution folder. For example:

```
$ scp <BF_INST_DIR>/distro/SRPMS/mlx-
OpenIPMI-2.0.25-0.g4fdc53d.src.rpm <ip-
address>:./<target_directory>/
```

If there are issues with building the OpenIPMI RPM, verify that the swig package is not installed.

```
$ yum remove -y swig
```

6. Generate a binary RPM from the ipmb-dev-int source RPM and install it. Run:

```
$ rpmbuild --rebuild ipmb-dev-int-1.0-0.g304ea0c.src.rpm
```

7. Generate a binary RPM from the `ipmb-host` source RPM and install it. Run:

```
$ rpmbuild --rebuild ipmb-host-1.0-0.g304ea0c.src.rpm
```

8. Load OpenIPMI, `ipmb-host`, and `ipmb-dev-int` RPM packages. Run:

```
$ yum install -y /root/rpmbuild/RPMS/aarch64/mlx-OpenIPMI-2.0.25-0.g581ebbb_5.4.0_49.el7a.aarch64.aarch64.rpm
$ yum install -y /root/rpmbuild/RPMS/aarch64/ipmb-dev-int-1.0-0.g304ea0c_5.4.0_49.el7a.aarch64.aarch64.rpm
$ yum install -y /root/rpmbuild/RPMS/aarch64/ipmb-host-1.0-0.g304ea0c_5.4.0_49.el7a.aarch64.aarch64.rpm
```

9. Load the IPMB driver. Run:

```
$ modprobe ipmb-dev-int
```

10. Install and start `rasdaemon` package. Run:

```
yum install rasdaemon
systemctl enable rasdaemon
systemctl start rasdaemon
```

11. Start the IPMI daemon. Run:

```
$ systemctl enable mlx_ipmid
$ systemctl start mlx_ipmid
$ systemctl enable set_emu_param
$ systemctl start set_emu_param
```

## 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