Intelligent Platform Management Interface

Table of contents

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^{2}C address 0x30 by default. The I^{2}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^{2}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^{2}C bus

BlueField BMC IPMI Commands

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

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

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

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

```
ipmitool -I ipmb <ipmitool command>
```

The following table provides a list of supported ipmitool command arguments:

Command Description	Ipmitool Command	Relevant IPMI 2.0 Rev 1.1 Spec Section
Get device ID	mc info	20.1
Broadcast “Get Device ID”	Part of "mc info"	20.9
Get BMC global enables	mc getenables	22.2
Get device SDR info	sdr info	35.2
Get device SDR	"sdr get", "sdr list" or "sdr elist"	35.3
Get sensor hysteresis	sdr get <sensor-id>	35.7
Set sensor threshold	sensor thresh <sensor-id> <threshold> <setting> • sensor-id – name of the sensor for which a threshold is to be set • threshold – which threshold to set ◦ ucr – upper critical ◦ unc – upper non-critical ◦ lnc – lower non-critical ◦ lcr – lower critical • setting – the value to set the threshold to To configure all lower thresholds, use : sensor thresh <sensor-id> lower <lnr> <lcr> <lnc>	35.8

Command Description	Ipmitool Command	Relevant IPMI 2.0 Rev 1.1 Spec Section
	Note The lower non-recoverable <lnr> option is not supported To configure all upper thresholds, use: sensor thresh <sensor-id> upper <unc> <ucr> <unr> **Note** The upper non-recoverable <unr> option is not supported	
Get sensor threshold	sdr get <sensor-id>	35.9
Get sensor event enable	sdr get <sensor-id>	35.11
Get sensor reading	sensor reading <sensor-id>	35.14
Get sensor type	sdr type <type>	35.16
Read FRU data	fru read <fru-number> <file-to-write-to>	34.2
Get SDR repository info	sdr info	33.9
Get SEL info	"sel" or "sel info"	40.2

Command Description	Ipmitool Command	Relevant IPMI 2.0 Rev 1.1 Spec Section
Get SEL allocation info	"sel" or "sel info"	40.3
Get SEL entry	"sel list" or "sel elist"	40.5
Add SEL entry	sel add <filename>	40.6
Delete SEL entry	sel delete <id>	40.8
Clear SEL	sel clear	40.9
Get SEL time	sel time get	40.1
Set SEL time	sel time set "MM/DD/YYYY HH:M:SS"	40.11

List of IPMI Supported Sensors

Sensor	ID	Description
bluefield_temp	0	Support NIC monitoring of BlueField's temperature
ddr0_0_temp [1]	1	Support monitoring of DDR0 temp (on memory controller 0)
ddr0_1_temp [1]	2	Support monitoring of DDR1 temp (on memory controller 0)
ddr1_0_temp [1]	3	Support monitoring of DDR0 temp (on memory controller 1)
ddr1_1_temp [1]	4	Support monitoring of DDR1 temp (on memory controller 1)
p0_temp	5	Port 0 temperature
p1_temp	6	Port 1 temperature
p0_link	7	Port0 link status
p1_link	8	Port1 link status

1. On BlueField-2 and BlueField-3 based boards, DDR sensors and FRUs are not supported. They will appear as no reading.

List of IPMI Supported FRUs

FRU	ID	Description
update_timer	0	set_emu_param.service is responsible for collecting data on sensors and FRUs every 3 seconds. This regular update is required for sensors but not for FRUs whose content is less susceptible to change. update_timer is used to sample the FRUs every hour instead. Users may need this timer in the case where they are issuing several raw IPMItool FRU read commands. This helps in assessing how much time users have to retrieve large FRU data before the next FRU update. update_timer is a hexadecimal number.
fw_info	1	NVIDIA® ConnectX® firmware information, Arm firmware version, and MLNX_OFED version. The fw_info is in ASCII format.
nic_pci_dev_info	2	NIC vendor ID, device ID, subsystem vendor ID, and subsystem device ID. The nic_pci_dev_info is in ASCII format.
cpuinfo	3	CPU information reported in lscpu and /proc/cpuinfo. The cpuinfo is in ASCII format.
ddr0_0_spd2	4	FRU for SPD MC0 DIMM 0 (MC = memory controller). The ddr0_0_spd is in binary format.
ddr0_1_spd2	5	FRU for SPD MC0 DIMM1. The ddr0_1_spd is in binary format.
ddr1_0_spd2	6	FRU for SPD MC1 DIMM0. The ddr1_0_spd is in binary format.
ddr1_1_spd2	7	FRU for SPD MC1 DIMM1. The ddr1_1_spd is in binary format.

FRU	ID	Description
emmc_info	8	eMMC size, list of its partitions, and partitions usage (in ASCII format). eMMC CID, CSD, and extended CSD registers (in binary format). The ASCII data is separated from the binary data with "StartBinary" marker.
qsfp0_eeprom	9	FRU for QSFP 0 EEPROM page 0 content (256 bytes in binary format)
qsfp1_eeprom	10	FRU for QSFP 1 EEPROM page 0 content (256 bytes in binary format)

ip_addresses	11	This FRU file can be used to write the BMC port 0 and port 1 IP addresses to the BlueField. It is empty to begin with.
		The file passed through the `ipmitool fru write 11 <file>` command must have the following format:
		`BMC: XXX.XXX.XXX.XXX`
		`P0: XXX.XXX.XXX.XXX`
		`P1: XXX.XXX.XXX.XXX`
		The size of the written file should be exactly 61 bytes.

dimms_ce_ue	12	FRU reporting the number of correctable and uncorrectable errors in the DIMMs. This FRU is updated once every 3 seconds.
eth0	13	Network interface 0 information. Updated once every minute.
eth1	14	Network interface 1 information. Updated once every minute.
bf_uid	15	BlueField UID
eth_hw_counters	16	List of ConnectX interface hardware counters

1. On BlueField-2 and BlueField-3 based boards, DDR sensors and FRUs are not supported. They will appear as no reading.

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>
```

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^{2}C Address
BlueField ipmb_dev_int	0x30
BMC ipmb_host	0x20

I2C Addresses for BlueField-initiated Requests

Device	I^{2}C Address
BlueField ipmb_host	0x11
BMC ipmb_dev_int	0x10

Changing I2C Addresses

To use a different BlueField or BMC I^{2}C address, you must make changes to the following files' variables.

Filename Path	Parameter Change

/usr/bin/set_emu_param.sh	The ipmb_dev_int and ipmb_host drivers are registered at the following I^{2}C addresses:
	• IPMB_DEV_INT_ADD=<BlueField I2C Address 1>
	• IPMB_HOST_ADD=<BlueField I2C Address 2>
	These addresses must be different from one another. Otherwise, one of the drives will fail to register.
	To change the BMC I^{2}C address:
	`IPMB_HOST_CLIENTADDR=<BMC I2C Address>`
	`<I2C Address> must be equal to: 0x1000+<7-bit I2C address>`

External Host IPMI Commands

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:

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
- eeprom.ko
- i2c-dev.ko

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

```
modprobe ee1004
modprobe at24
modprobe eeprom
```

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/ipmitool-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 -sfn /usr/lib/libi2c.so.0.1.1 /lib64/libi2c.so
ln -sfn /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
```

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
```