AmpereOne™ 64-Bit Multi-Core Processor

Ampere® OpenBMC User's Manual

April 19, 2024

Document Issue 0.90

Contents

Contents (continued)

Contents (continued)

Figures

Tables

1. Overview

This user's manual provides user guidelines and information about using Ampere® OpenBMC firmware for Ampere reference platforms.

1.1 Scope

This document provides guidelines only for BMC features that are required for Ampere reference platforms. Other common features might or might not work but are out of scope for this document.

Not all hardware configurations for Ampere reference platforms are supported in Ampere OpenBMC. The following table shows the supported platforms and their supported hardware configurations.

Table 1: Supported platforms

PLATFORMS	PLATFORM NAME	SUPPORTED HARDWARE CONFIGURATION
Mt.Jade	mtjade	Mt.Jade 2U
Mt.Mitchell	mtmitchell-dcscm	Mt.Mitchell DC-SCM 2U 24SFF SKU

1.2 Supported Pluggable Devices

Table 2 lists the supported pluggable hardware devices.

Table 2: List of Supported Pluggable Hardware Devices

TYPES	SUPPORTED MODELS	NOTE
Power Supply	Great Wall GW-CRPS2000DW	—
	Delta DPST-2030AB	—
	Artesyn CSU2000AP	—
	LiteOn PS-2162-12L6	—
	Chicony R18-2K0P1XA	—
	Chicony R18-1K3P1XA	—
OCP	Mellanox Technologies MT27800 Family [ConnectX-5]	—
	Broadcom 25G 4P 57504 OCP Mezz	—

1.3 User Interfaces

1.3.1 External User Interfaces

Users can access the BMC using WebUI, Intelligent Platform Management Interface (IPMI), and Redfish.

1.3.2 In-Band IPMI

Users can also run IPMI from the Host OS with root privileges (or with sudo) using the IPMI SMBus System Interface (SSIF).

1.3.3 BMC Console

End users must not have access to the BMC console. This interface is intended only for developers and manufacturing engineers. BMC console logins are limited to users with a deep understanding of the system.

To update firmware from the BMC console, download the firmware images into the **/tmp** folder. Users must ensure enough free storage (at least 20 MB) for the system to work. Filling the root file system or **/tmp** folder full or almost full can cause firmware updates to fail. Avoid writing data on flash storage.

1.4 Ampere OpenBMC Revisions

1.4.1 Naming Conventions

Table 3 shows the naming conventions used for BMC revisions.

Table 3: Ampere OpenBMC Release Versioning

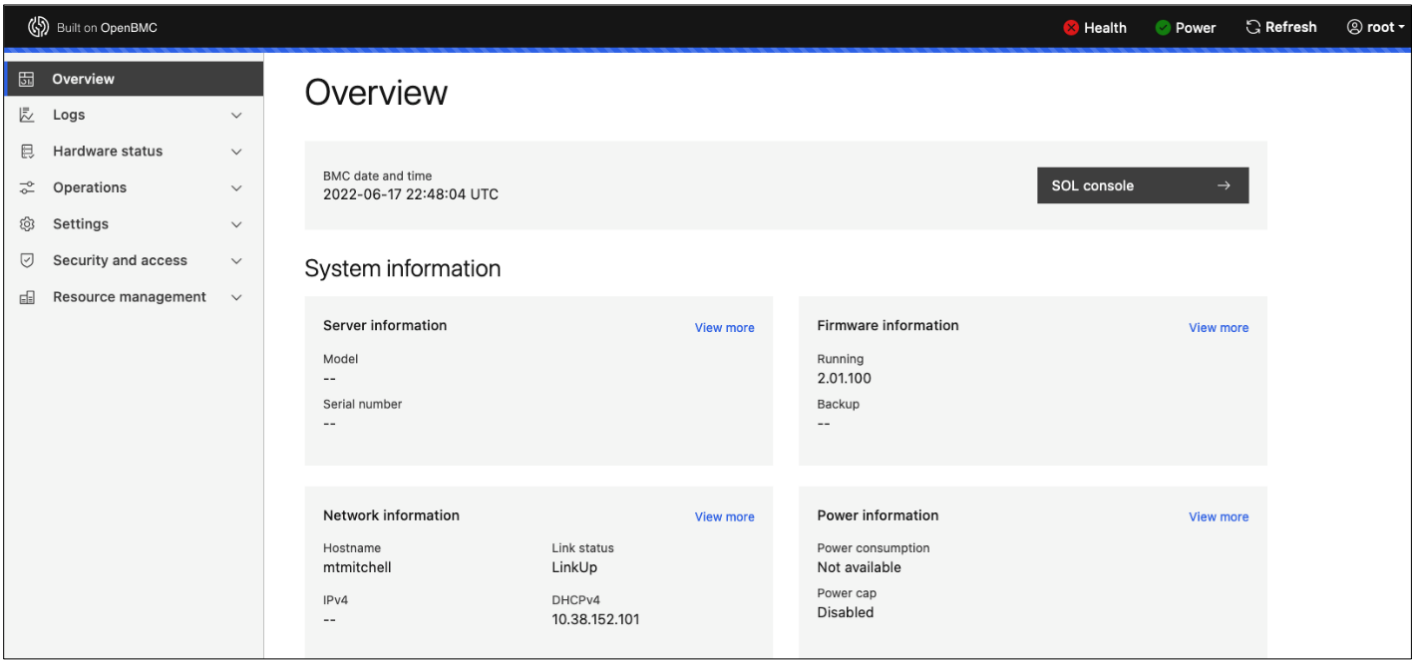
IDENTIFIER	DESCRIPTION
REVISION	MM.NN.XXX | | |- BUILD_NUMBER | |- MINOR VERSION |- MAJOR VERSION
MAJOR VERSION	Major version
MINOR VERSION	Include two numbers, from 00-99. This field will be increased by each release, reset to zero when major version is increased.
BUILD_NUMBER	100, 200, and so on

1.4.2 Revision Reports

Revision information is displayed differently in the external user interfaces.

1. WebUI

Figure 1: WebUI Interface



Notes:

- Running: The image is being activated.
- Backup: The image is being updated during the updating progress.

2. IPMI `mc info` command (the build number is not displayed).

```
# ipmitool mc info
Device ID                 : 32
Device Revision           : 1
Firmware Revision         : 2.01
```

Note: Auxiliary information is NOT supported.

3. Redfish manager schema (Firmware version field).

```
➜  ~ curl -k -H "X-Auth-Token: $token" -X GET https://${bmcip}/redfish/v1/Managers/bmc | grep
       FirmwareVersion
  "FirmwareVersion": "2.10.100",
```

1.5 Firmware download

Ampere OpenBMC source codes are published at https://github.com/ampere-openbmc/openbmc.

People can check out for specific release using the following commands:

```
$ git clone https://github.com/ampere-openbmc/openbmc.git
$ cd openbmc
$ git checkout <release tag>
```

Where release tags have format of v<version>-ampere, for example, v2.13.100-ampere.

To compile firmware image for a supported platform, using the following example with mtmitchell-dcscm:

```
$ . setup mtmitchell-dcscm
$ bitbake obmc-phosphor-image
```

The built image will be available in ./build/mtmitchell-dcscm/tmp/deploy/images/mtmitchell-dcscm/

Note: Prerequisites might be needed based on the OS being compiled as mentioned in the README.md in the Ampere OpenBMC Github repository.

2. User Accounts

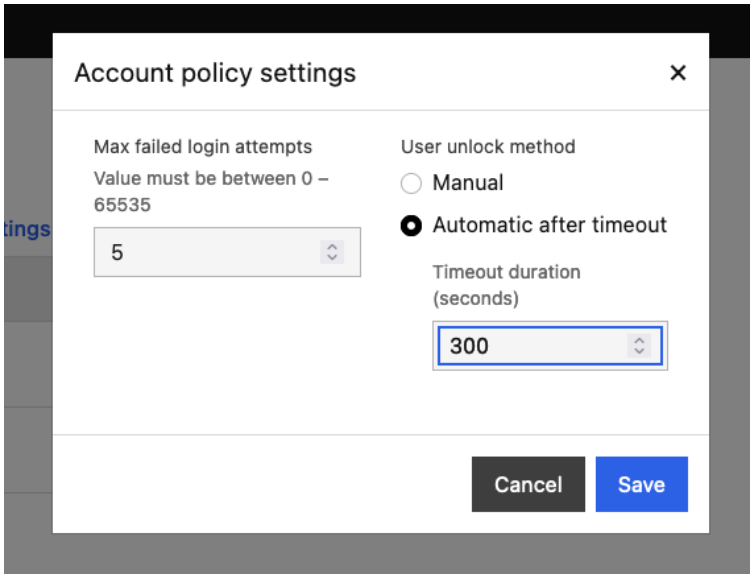
2.1 Default Account

The BMC uses **root/0penBmc** as the default account for BMC console login, WebUI, Rest/Redfish, and IPMI.

2.2 Account Policy Settings

The BMC supports an account policy settings feature which can lock out users after too many failed login attempts. These settings can be configured using WebUI at **Security and access > User Management > Account policy settings**.

Figure 2: Account Policy Settings



Note: Account policy settings do not apply to the "root" account.

2.3 Ampere Password Policy

Ampere provides additional password requirements listed as follows on top of the default OpenBMC password policy (through libpam library):

- A password must have at least nine characters.
- A password must include at least one lowercase letter, one uppercase letter, one digit, and one special character.
- A password must not include more than three consecutive identical characters.

The following are some (but not all) password patterns that libpam rejects:

- Include its username in straight and reverse forms.
- <word>xxx where <word> is a word in dictionary like Ampere, dictionary, and xxx are any three characters.
- Monotonic sequence with:
  - One monotonic sequence with length > 5 (that is, 123456, abcdef, and so on)
  - Two adjacent monotonic sequences with total length > 6 (that is, abcd123, abc4567, and so on)

Note: Identifying all the policies that OpenBMC/libpam rejects is out of scope.

3. Firmware Updates

Ampere OpenBMC supports firmware updates using Redfish (SimpleUpdate) and WebUI (calls Redfish API SimpleUpdate) for these components: BMC firmware and Host firmware.

However, Ampere OpenBMC supports MANIFEST files with ExtendedVersion fields which Ampere OpenBMC uses to specify additional firmware types so that the backend script can call appropriate commands to flash the firmware image on appropriate hardware devices, such as EEPROMs, Complex Programmable Logic Device (CPLDs), and so on.

3.1 BMC Firmware Updates

3.1.1 Recover BMC Firmware from U-boot

If the BMC firmware is not programmed for SPI-NOR flash memory on the platform board, or if BMC cannot boot, an external programmer (such as a DediProg) is required to program the BMC firmware. Refer to SPI-NOR and reference documentation for programming information.

If the BMC fails to boot Linux, or if Linux boots but the software stack does not work but still reaches U-boot, users must follow the steps in this section to flash the BMC firmware using U-boot.

When a user powers-up or resets the BMC, the U-boot boot loader starts running on the BMC. After a few seconds, U-boot automatically boots the pre-installed kernel. To stop booting at U-boot, the user must repeatedly press a key on the serial console connected to the BMC UART. When U-boot stops, it displays a command line console with the prompt `ast#`.

From the U-boot command line, users can configure the network as follows:

1. Disable the watchdog.

   ```
   # mw 0x1e78502c 0
   ```

2. Set the Active Ethernet Interface to eth1.

   ```
   ast# setenv ethact FTGMAC100#1
   ```

3. Get the IP address from the Dynamic Host Control Protocol (DHCP) server.

   ```
   ast# dhcp
   ```

4. Set the Trivial File Transfer Protocol (TFTP) server IP address.

   ```
   ast# set serverip <TFTP server IP address>
   ```

5. Download the BMC firmware (in mtd format) at address 0x80000000.

   ```
   ast# tftp 0x80000000 <BMC static.mtd from TFTP server>
   ```

6. Flash the OpenBMC firmware into SPI-NOR:
   - With the BMC firmware using u-boot 2016.07, using one of the following commands:
     - Run the following command to update the BMC firmware for the MAIN SPI-NOR device.

     ```
     ast# protect off 0x20000000 +${filesize}; erase 0x20000000 +${filesize}; cp.b 0x80000000
     0x20000000 ${filesize};protect on all
     ```

     - Run the following command to update the BMC firmware for the FAILOVER SPI-NOR device as desired.

     ```
     ast# protect off 0x24000000 +${filesize}; erase 0x24000000 +${filesize}; cp.b 0x80000000
     0x24000000 ${filesize};protect on all
     ```

   - With the BMC firmware using u-boot 2019.04, using one of these commands:
     - Run this command to update the BMC firmware for the MAIN SPI-NOR device.

     ```
     ast # sf probe 0; sf protect unlock 0 $filesize;sf erase 0 $filesize;
     sf write 0x80000000 0 $filesize
     ```

- Run this command to update the BMC firmware for the FAILOVER SPI-NOR device as desired.

```
ast# sf probe 1; sf protect unlock 0 $filesize;sf erase 0 $filesize;
sf write 0x80000000 0 $filesize
```

7. Reset the BMC and boot with the new image.

```
ast# reset
```

3.1.2 Update BMC Firmware from Linux

OpenBMC supports flashing any BMC image in raw format from the BMC console and flashes the image into the BMC SPI-NOR without examining the contents.

To flash a raw image from the OpenBMC console:

1. Copy the raw image to the BMC and rename it to /run/initramfs/image-bmc.

```
# scp 10.38.64.31:~/dl/mtmitchell_openbmc_2.01.100.static.mtd
/run/initramfs/image-bmc
```

2. Reboot to trigger the flash.

```
# reboot
```

3. After flashing finishes, the BMC reboots to the new firmware.

3.1.3 Update Firmware using Redfish and WebUI

3.1.3.1 Prepare the BMC Firmware Package for Upgrade

The BMC supports firmware to flash in *.tar format. This includes images to flash and a MANIFEST file to contain firmware information. Users must prepare the *.tar package.

To flash the secondary SPI-NOR BMC firmware, users must update the MANIFEST file to set/add ExtendedVersion=alternate. The MANIFEST file contents resemble this example:

```
purpose=xyz.openbmc_project.Software.Version.VersionPurpose.BMC
version=2.01.100
KeyType=OpenBMC
HashType=RSA-SHA256
MachineName=<platform name>
ExtendedVersion=alternate
```

Notes:

1. The BMC does not support flashing the same firmware image with the running firmware, including flashing to a different BMC SPI-NOR (main vs. failover SPI-NOR).

3.1.3.2 Update BMC Firmware using Redfish

The BMC firmware image is flashed using the following Redfish commands, where <image file path> is the path to the *.tar image:

```
$ token=`curl -k -H "Content-Type: application/json" -X POST https://${BMC_IP}/login -d
       '{"username" :  "root", "password" :  "0penBmc"}' | grep token | awk '{print $2;}' | tr
       -d '"'`
$ uri=$(curl -k -H "X-Auth-Token: $token" https://${BMC_IP}/redfish/v1/UpdateService | jq -r
       ' .HttpPushUri')
$ curl -k -H "X-Auth-Token: $token" -H "Content-Type: application/octet-stream" -X POST -T
       <image file path> https://${BMC_IP}${uri}
```

See https://github.com/openbmc/docs/blob/master/REDFISH-cheatsheet.md#firmware-update for more information.

3.1.3.3 Use WebUI to Flash the Firmware

Perform the following steps to support flashing the BMC firmware image using WebUI:

1. Boot the BMC and enter WebUI using the Administrator privilege username/password.
2. Navigate to **Configuration > Firmware page >** choose firmware image in *.tar format.

Figure 3: Firmware Page

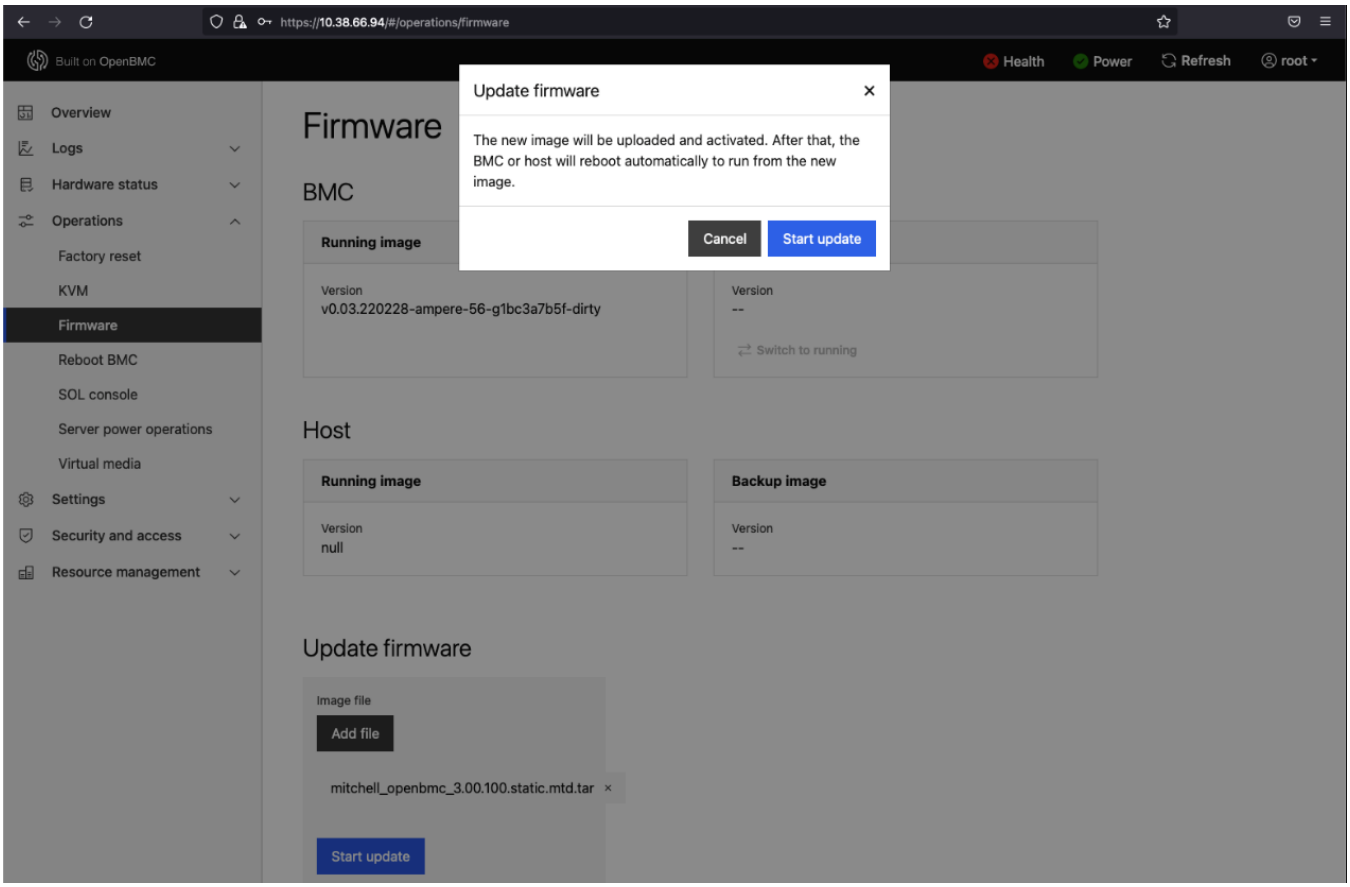
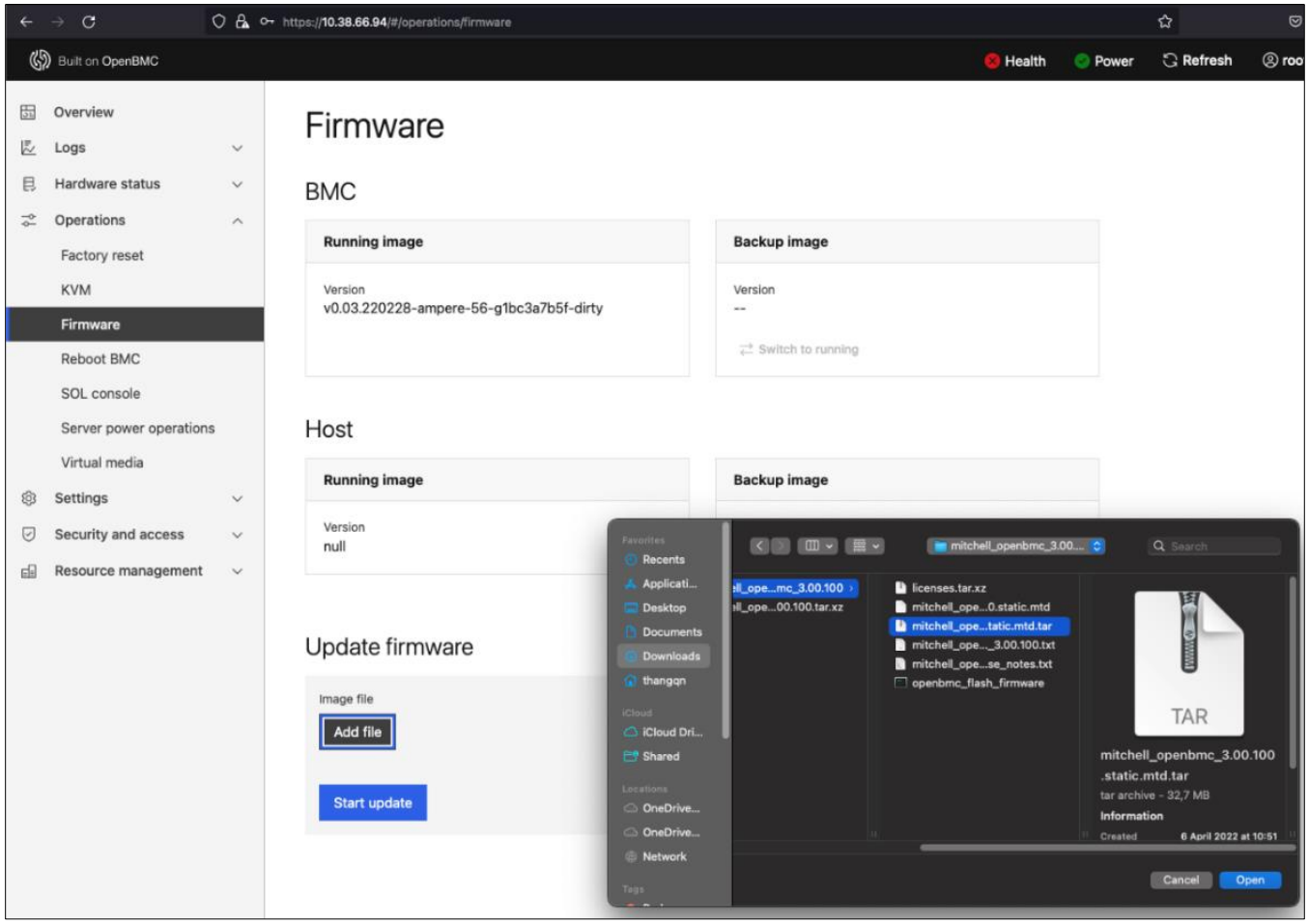


Figure 4: Choose Firmware Image



3. Click **Add file** and select the firmware image to upload.
4. The BMC firmware will be uploaded, and the update is activated. The BMC automatically reboots after the BMC firmware update.

3.2 Host and Device Firmware Updates

3.2.1 Update Host and Device Firmware using the BMC Console

Perform the following steps to flash firmware using the BMC console:

1. Copy the firmware image to the **/tmp** folder using either scp or tftp. For example:

```
# scp thangqn@10.38.153.22:/tftpboot/edk2/spinor.img /tmp/
```

2. Run one of the following commands to flash the desired firmware image:
   - Flash host firmware into the main Host SPI-NOR:

     ```
     # ampere_flash_bios.sh <image>
     ```

   - Flash host firmware into the secondary Host SPI-NOR (if present):

     ```
     # ampere_flash_bios.sh <image> 2
     ```

   - Flash BSD/EEPROM firmware into the main Boot EEPROM:

     ```
     # ampere_firmware_upgrade.sh eeprom <image>
     ```

- Flash FRU binary image to the motherboard FRU EEPROM:

  ```
  # ampere_firmware_upgrade.sh fru <fru.bin>
  ```

- Flash FRU binary image to the BMC FRU EEPROM (if present):

  ```
  # ampere_firmware_upgrade.sh fru <fru.bin> 2
  ```

- Flash MB CPLD firmware (using JTAG):

  ```
  # ampere_firmware_upgrade.sh mb_cpld <image.jed>
  ```

- Flash BMC CPLD firmware (using JTAG):

  ```
  # ampere_firmware_upgrade.sh bmc_cpld <image.jed>
  ```

- Flash backplane CPLD firmware (using I2C):

  ```
  # ampere_firmware_upgrade.sh bp_cpld <image.jed> <id>
  ```
  Where <id> follows backplane indexes for each supported platform.

Perform the following steps to check the firmware revision:

1. Check MB CPLD firmware.

   ```
   # ampere_firmware_version.sh mb_cpld
   ```

2. Check backplane CPLD firmware.

   ```
   # ampere_firmware_version.sh bp_cpld <id>
   ```
   where <id> follows backplane indexes for each supported platform.

The following table defines backplanes indexes for Mt.Mitchell platforms:

Table 4: Backplane Index for Mt. Mitchell Platform

INDEX	DEFINITION	NOTE
1	Front backplane 1	The Host must be powered ON before flashing backplane CPLD firmware
2	Front backplane 2	
3	Front backplane 3	
4	Rear backplane 1	
5	Rear backplane 2	

Note:

1. For the mainboard CPLD, an AC power cycle is required for the updated firmware to work.

3.2.2 Update Host and Device Firmware using Redfish and WebUI

3.2.2.1 Host Firmware Package

Host firmware components are packaged in tar files comprising the raw firmware image and a MANIFEST file to specify what the component is. The following is the MANIFEST file format:

```
purpose=xyz.openbmc_project.Software.Version.VersionPurpose.Host
version=<Revision>
ExtendedVersion=<ExtendedVersion>
KeyType=OpenBMC
HashType=RSA-SHA256
MachineName=<platform name>
```

Table 5 lists the values of Purpose and ExtendedVersion for various firmware components.

Table 5: Purpose and ExtendedVersion in MANIFEST File

FIRMWARE COMPONENT	EXTENDEDVERSION	SUPPORTED FILE EXTENSION	NOTES
Host UEFI firmware – main SPI-NOR	primary	img, rom	—
Host UEFI firmware – secondary SPI-NOR	secondary	img, rom	—
BSD/EEPROM – main Boot EEPROM	eeprom	slim, bin	Also use for Altra SCP firmware
BSD/EEPROM – secondary Boot EEPROM	eeprom-secondary	slim, bin	Also use for Altra SCP firmware
MB FRU	mbfru	bin	—
BMC FRU	bmcfru	bin	—
MB CPLD	mbcpld	jed, bin	—
BMC CPLD	bmccpld	jed, bin	—
BP CPLD X (X = 1, 2, …)	bpcpldX	jed, bin	—

Notes:

1. Although the BMC supports flashing an FRU binary into the FRU EEPROM, it is **not recommended** to change the FRU EEPROM. The FRU EEPROM contains board information from the manufacturer. If users flash FRU EEPROM for test purposes, the original FRU binary must be flashed again.
2. Images in the *.tar file must have correct file extensions for the corresponding firmware type.
3. OpenBMC uses the firmware revision from the MANIFEST file in the version field. Any string contained in this field is treated as firmware revision.

The following example shows the command to create a *.tar file for the firmware update:

```
# tar cf mbfru.tar MANIFEST mbfru.bin
```

3.2.2.2 Flash Host Firmware using Redfish

The steps described in *Section 3.1.3, Update Firmware using Redfish and WebUI* and *Section 3.1.3.3, Use WebUI to Flash the Firmware* can be used for Host firmware updates.

4. Host Power Control

The BMC supports multiple methods to control Host power, listed as follows:

1. Using WebUI: **Control > Server power operations**
2. Using IPMI: https://github.com/openbmc/docs/blob/master/IPMITOOL-cheatsheet.md#power
3. Using Redfish: https://github.com/openbmc/docs/blob/master/REDFISH-cheatsheet.md#host-power
4. From the Host OS using commands such as reboot, shutdown, and so on.
5. Using the power and reset buttons.

This chapter describes the supported actions and their behaviors for host power control.

4.1 Control Power using IPMI, WebUI, and Redfish

Table 6 lists the BMC behavior in supporting Host power control using Redfish, IPMI, and WebUI.

Table 6: Power Control from IPMI, WebUI, and Redfish

REDFISH	IPMI	WEBUI	ACTION
ForceOff	power down	Shutdown Server: Immediately	DC power switches OFF host immediately.
ForceOn	power up	Power on	DC power switches ON host immediately.
On	power up	Power on	Same as ForceOn
ForceRestart	hard reset	Reset Server: Immediately	Trigger SYS_RESET pin.
GracefulShutdown	soft off	Shutdown Server: Orderly	Gracefully shuts down the Host. Do nothing when the Host is currently switched OFF.
GracefulRestart	N/A	Reset Server: Orderly	Gracefully shuts down the Host, then triggers SYS_RESET pin.
PowerCycle	power cycle	N/A	Host ON: Gracefully shuts down (DC power is switched OFF if the Host does not have ACPI power enabled) then DC power is switched ON. Host OFF: Do nothing.

Note: Complete a Host power control command before triggering a new one. Sending new power control commands while a preceding command is in progress might result in unintended BMC behavior.

Note: Mt.Mitchell specific behavior:

- Redfish's ForceRestart, GracefulRestart and IPMI power reset:
  - When the Host is currently switched ON, gracefully shut down (DC power OFF if Host does not have ACPI power enabled) then DC power must be switched ON.
  - When the Host is currently switched OFF, power must be switched ON.
- Redfish's PowerCycle and IPMI power cycle: when the Host is currently switched OFF, turn the Host switch ON.

4.2 Control Power from Host OS

Table 7 lists how OpenBMC processes Host state control from the Host OS.

Table 7: Power Control from Host OS

HOST	ACTION
reboot	Waiting for the REBOOT_ACK pin then triggers SYS_RESET pin to reset the Host
halt	Gracefully shuts down and halts the Host; DC power remains turned ON
shutdown/power off	Gracefully shuts down the Host; DC power is turned OFF

Note: Mt.Mitchell specific behavior for Host reboot: Graceful shutdown, DC power is switched OFF, then the DC power is switched ON.

4.3 Power Control Restriction

During the firmware update operation using Redfish or WebUI, the following operations are prohibited:

- IPMI: chassis power, power on/reset/cycle.
- Redfish Chassis Power Control: ForceOn, On, ForceRestart, GracefulRestart, PowerCycle.
- Press the power button.

The BMC ignores the preceding operations when it is flashing firmware.

Note: This feature is not applicable for firmware update using engineering tools like **ampere_flash_bios.sh** and **ampere_firmware_upgrade.sh**.

5. Sensor Monitoring

OpenBMC supports sensor reporting using IPMI, WebUI, and Redfish.

The following are the points to be noted on sensor report:

- Sensor information (name, thresholds, and so on) are read only at the time the sensors are created. Any change in sensor source (for example, any change in Mpro for sensors) does not take effect until the next time the sensors are deleted and created.
- Ampere platform hardware does not support the presence pins for NVMe devices. The NVMe temperature sensors are not added again (after it is removed as the drivers or NVME backplane were unplugged previously) after NVMe backplanes or drives are hot plugged. The OpenBMC supports rescan for changes every five minutes. So, after unplugging and replugging, the NVMe sensors are expected to be available again after five minutes.

5.1 Sensor Monitoring using IPMI

The BMC supports report sensor information using standard IPMI commands like SDR list and sensor list.

For example:

```
# ipmitool sdr list
S0_0V8_D2D       | 0.83 Volts        | ok
S0_0V85_SOC      | 0.84 Volts        | ok
S0_0V85_RC_DDR0  | 0.87 Volts        | ok
S0_0V85_RC_DDR1  | 0.87 Volts        | ok
S0_0V9_RC5A      | 0 Volts           | ok
S0_1V8_RC5A      | 0 Volts           | ok
S0_0V75_PCP      | 0.74 Volts        | ok
S0_1V1_VDDQ0123  | 1.10 Volts        | ok
S0_1V1_VDDQ4567  | 0 Volts           | ok
S0_1V8_SOC       | 0.21 Volts        | ok
S0_1V2_SOC       | 0.02 Volts        | ok
S0_EXT_VREF      | 0.69 Volts        | ok
S1_0V8_D2D       | 0.02 Volts        | ok
S1_0V85_SOC      | 0 Volts           | ok
S1_0V85_RC_DDR0  | 0 Volts           | ok
S1_0V85_RC_DDR1  | 0.87 Volts        | ok
```

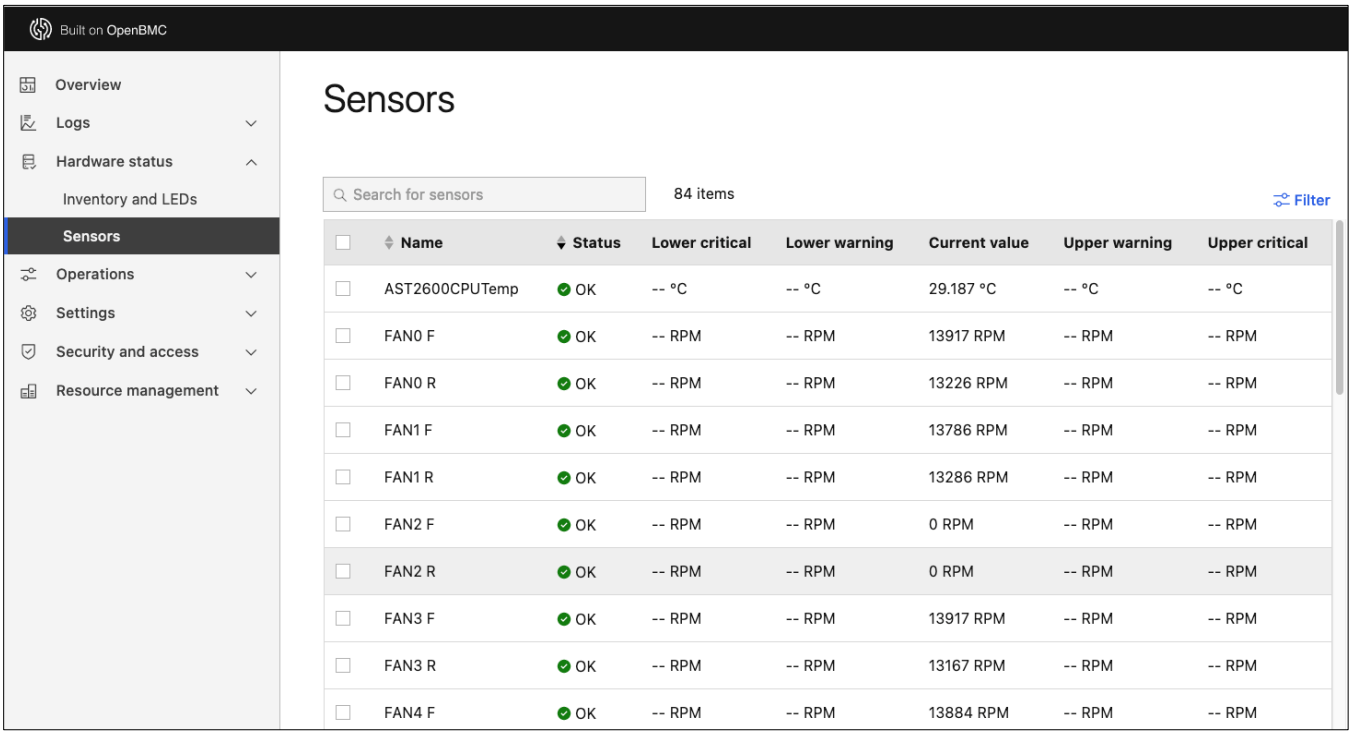
Notes:

- Disabled sensors are not hidden in `ipmitool sdr` or `ipmitool sensor` command output, but are displayed as "disabled" or "n/a."
- OpenBMC does not support these IPMI sensor attributes: UNC, LNC, SI, SC, AM, DM, and TM.

5.2 Sensor Monitoring using WebUI

The BMC displays sensor information with the threshold values using WebUI at **Heath > Sensors**.

Figure 5: Sensors



5.3 Sensor Monitoring using Redfish

Sensors are reported in Redfish under the Sensor schema at /redfish/v1/Chassis/<board>/Sensors where <board> is defined as follows:

Table 8: Redfish chassis instance names

PLATFORM	<BOARD>	DESCRIPTION
Mt.Jade	Mt_Jade	All sensors are available for Mt.Jade
Mt.Mitchell	Mt_Mitchell_Motherboard	On-board and virtual sensors on the Mt.Mitchell motherboard
	Mt_Mitchell_DCSCM_BMC	On-board sensors on the Mt.Mitchell BMC board
	Mitchell_BP_X	All sensors on the backplane X
	chassis	Host CPU sensors

Example: list out sensors under Mt_Mitchell_Motherboard chassis instance and read a sensor:

```
$ curl -X GET --user root:0penBmc --insecure \
       -H "Content-Type: application/json"   \
       https://10.76.204.196/redfish/v1/Chassis/Mt_Mitchell_Motherboard/Sensors
{
  "@odata.id": "/redfish/v1/Chassis/Mt_Mitchell_Motherboard/Sensors",
  "@odata.type": "# SensorCollection.SensorCollection",
  "Description": "Collection of Sensors for this Chassis",
  "Members": [
    {
      "@odata.id": "/redfish/v1/Chassis/Mt_Mitchell_Motherboard/Sensors/current_PSU0_IINPUT"
    },
    {
      "@odata.id": "/redfish/v1/Chassis/Mt_Mitchell_Motherboard/Sensors/current_PSU0_IOUTPUT"
    },
  .....
  ],
  "Members@odata.count": 85,
  "Name": "Sensors"
}
$ curl -X GET --user root:0penBmc --insecure \
       -H "Content-Type: application/json"  \
https://10.38.152.101/redfish/v1/Chassis/Mt_Mitchell_Motherboard/Sensors/current_PSU0_IINPUT
{
  "@odata.id": "/redfish/v1/Chassis/Mt_Mitchell_Motherboard/Sensors/current_PSU0_IINPUT",
  "@odata.type": "#Sensor.v1_2_0.Sensor",
  "Id": "current_PSU0_IINPUT",
  "Name": "PSU0 IINPUT",
  "Reading": 2.933,
  "ReadingType": "Current",
  "ReadingUnits": "A",
  "Status": {
    "Health": "OK",
    "State": "Enabled"
  }
}
```

6. Platform Level Data Model (PLDM)

Note: Mt.Jade does not support PLDM, therefore, it is not applicable for all features mentioned in this chapter.

6.1 IPMI RAS Error Report

The BMC reports all the RAS Error events for CPU Core, MCU, PCIe, and SoC using IPMI SEL in accordance with the *AmpereOne BMC System Event Log (SEL) Specification*.

6.2 Redfish RAS Error Report

The BMC supports Redfish reporting for RAS Error following the SBMR 2.0 Specification and OpenBMC FaultLog description. Whenever the Host sends a RAS error by PLDM, the BMC creates a new FaultLog event which users can get using Redfish.

```
$ curl -X GET --user root:0penBmc -H "Content-Type: application/json" -H "If-match: " --
       insecure https://10.6.244.87/redfish/v1/Managers/bmc/LogServices/FaultLog/Entries
{
  "@odata.id": "/redfish/v1/Managers/bmc/LogServices/FaultLog/Entries",
  "@odata.type": "#LogEntryCollection.LogEntryCollection",
  "Description": "Collection of FaultLog Dump Entries",
  "Members":
    {
      "@odata.id": "/redfish/v1/Managers/bmc/LogServices/FaultLog/Entries/1",
      "@odata.type": "#LogEntry.v1_9_0.LogEntry",
      "AdditionalDataURI":
       "/redfish/v1/Managers/bmc/LogServices/FaultLog/Entries/1/attachment",
      "Created": "2023-03-01T09:05:15.417923+00:00",
      "DiagnosticDataType": "CPER",
      "EntryType": "Event",
      "Id": "1",
      "Name": "FaultLog Dump Entry"
    },
```

A link is contained inside the FaultLog entry for users to download CPER data:

```
$ curl -X GET --user root:0penBmc -H "Content-Type: application/json" -H "If-match: " --
       insecure
       https://10.76.244.87/redfish/v1/Systems/system/LogServices/FaultLog/Entries/1/attachmen
       t --output cper.dump
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   344  100   344    0     0    369      0 --:--:-- --:--:-- --:--:--   369
$ hexdump cper.dump
0000000 5043 5245 0101 ffff ffff 0001 0002 0000
0000010 0000 0000 0158 0000 0000 0000 0000 0000
0000020 0000 0000 0000 0000 0000 0000 0000 0000
*
0000050 d5ac 09a9 5204 4214 e596 9994 752e cd2b
0000060 0000 0000 0000 0000 0000 0000 0000 0000
*
0000080 00c8 0000 0090 0000 0100 0000 0000 0000
0000090 3d16 e19e bc11 11e4 aa9c 05c2 5d1d b046
00000a0 0000 0000 0000 0000 0000 0000 0000 0000
00000b0 0002 0000 0000 0000 0000 0000 0000 0000
00000c0 0000 0000 0000 0000 0001 0000 0001 0000
00000d0 0090 0000 0000 0000 0000 0000 0000 0000
```

```
00000e0 0000 0000 0000 0000 0000 0000 0000 0000
00000f0 6800 0000 0000 0000 0000 0000 0000 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000120 0000 0000 0000 0000 0000 4100 0000 0000
0000130 0000 0000 0000 0000 0000 0000 0000 0000
*
0000158
```

Then, libcper can be used to convert CPER data in binary format to json format:

```
$ ./cper-convert to-json cper.dump
{
  "header":{
    "revision":{
      "major":1,
      "minor":1
    },
    "sectionCount":1,
    "severity":{
      "code":2,
      "name":"Corrected"
    },
    "validationBits":{
      "platformIDValid":false,
      "timestampValid":false,
      "partitionIDValid":false
    },
    "recordLength":344,
    "creatorID":"00000000-0000-0000-0000000000000000",
    "notificationType":{
      "guid":"09a9d5ac-5204-4214-96e594992e752bcd",
      "type":"PEI"
    },
    "recordID":0,
    "flags":{
      "value":0,
      "name":"Unknown"
    },
    "persistenceInfo":0
  },
  "sectionDescriptors":[
    {
      "sectionOffset":200,
      "sectionLength":144,
      "revision":{
        "major":1,
        "minor":0
      },
      "validationBits":{
        "fruIDValid":false,
        "fruStringValid":false
      },
      "flags":{
        "primary":false,
        "containmentWarning":false,
```

```
        "reset":false,
        "errorThresholdExceeded":false,
        "resourceNotAccessible":false,
        "latentError":false,
        "propagated":false,
        "overflow":false
      },
      "sectionType":{
        "data":"e19e3d16-bc11-11e4-9caac2051d5d46b0",
        "type":"ARM"
      },
      "severity":{
        "code":2,
        "name":"Corrected"
      }
    }
  ],
  "sections":[
    {
      "validationBits":{
        "mpidrValid":true,
        "errorAffinityLevelValid":false,
        "runningStateValid":false,
        "vendorSpecificInfoValid":false
      },
      "errorInfoNum":1,
      "contextInfoNum":0,
      "sectionLength":144,
      "errorAffinity":{
        "value":0,
        "type":"Vendor Defined"
      },
      "mpidrEl1":0,
      "midrEl1":0,
      "running":false,
      "psciState":0,
      "errorInfo":[
        {
          "version":0,
          "length":104,
          "validationBits":{
            "multipleErrorValid":false,
            "flagsValid":false,
            "errorInformationValid":false,
            "virtualFaultAddressValid":false,
            "physicalFaultAddressValid":false
          },
          "errorType":{
            "value":0,
            "name":"Cache Error"
          },
          "multipleError":{
            "value":0,
            "type":"Single Error"
```

```
                },
                "flags":{
                  "firstErrorCaptured":false,
                  "lastErrorCaptured":false,
                  "propagated":false,
                  "overflow":false
                },
                "errorInformation":{
                  "validationBits":{
                    "transactionTypeValid":false,
                    "operationValid":false,
                    "levelValid":false,
                    "processorContextCorruptValid":false,
                    "correctedValid":false,
                    "precisePCValid":false,
                    "restartablePCValid":false
                  },
                  "transactionType":{
                    "value":0,
                    "name":"Instruction"
                  },
                  "operation":{
                    "value":0,
                    "name":"Generic Error"
                  },
                  "level":0,
                  "processorContextCorrupt":false,
                  "corrected":false,
                  "precisePC":false,
                  "restartablePC":false
                },
                "virtualFaultAddress":0,
                "physicalFaultAddress":0
              }
            ],
            "contextInfo":[
            ],
            "vendorSpecificInfo":{
              "typeId":0,
              "subTypeId":0,
              "instanceId":0,

             "remainData":"AAAAAAAAAAAAAAAAAAAAAAAAAEEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
             AAAAAAAAAAAAA=="
            }
          }
        ]
      }
```


To avoid running out of space, the BMC only keeps the latest 100 FaultLog entries and the latest 10 CPER data.

When the number of CPER entries exceed 10, the oldest CPER data is deleted, but the CPER entries are kept and the AdditionalDataURI field is hidden for FaultLog entries without CPER data.


```
{
  "@odata.id": "/redfish/v1/Managers/bmc/LogServices/FaultLog/Entries/1",
```

```
    "@odata.type": "#LogEntry.v1_9_0.LogEntry",
    "Created": "2023-03-01T08:48:35.121558+00:00",
    "DiagnosticDataType": "CPER",
    "EntryType": "Event",
    "Id": "1",
    "Name": "FaultLog Dump Entry",
  },
```


When the number of FaultLog entries exceeds 100, the oldest FaultLog entry is deleted (that is, you can no longer see the deleted FaultLog entry through Redfish).

Notes:

- Upgrading BMC firmware clears FaultLog and CPER data.
- FaultLog contains CPER and Crashdump entries.
- BERT and Diagnostic are OEM type of Crashdump.

6.3 BERT Crash Dump Report

The BMC supports Redfish reporting for BERT crash dump data in the same FaultLog entry as RAS CPER data, except the DiagnosticDataType for the crash dump is "OEM" and OEMDiagnosticDataType is "BERT." The maximum number of BERT records is two. When the number of BERT entries exceed two, the oldest BERT data is deleted, but the BERT entry is kept.

For example:

```
$ curl -X GET --user root:0penBmc -H "Content-Type: application/json" -H "If-match: " --
       insecure https://10.76.244.87/redfish/v1/Managers/bmc/LogServices/FaultLog/Entries
{
  "@odata.id": "/redfish/v1/Managers/bmc/LogServices/FaultLog/Entries",
  "@odata.type": "#LogEntryCollection.LogEntryCollection",
  "Description": "Collection of FaultLog Dump Entries",
  "Members": [
    {
      "@odata.id": "/redfish/v1/Managers/bmc/LogServices/FaultLog/Entries/1",
      "@odata.type": "#LogEntry.v1_9_0.LogEntry",
      "Created": "2023-03-01T08:48:35.121558+00:00",
      "DiagnosticDataType": "OEM",
      "EntryType": "Event",
      "Id": "1",
      "Name": "FaultLog Dump Entry",
      "OEMDiagnosticDataType": "BERT"
    },
```

A link is available inside the FaultLog entry for users to download BERT data:

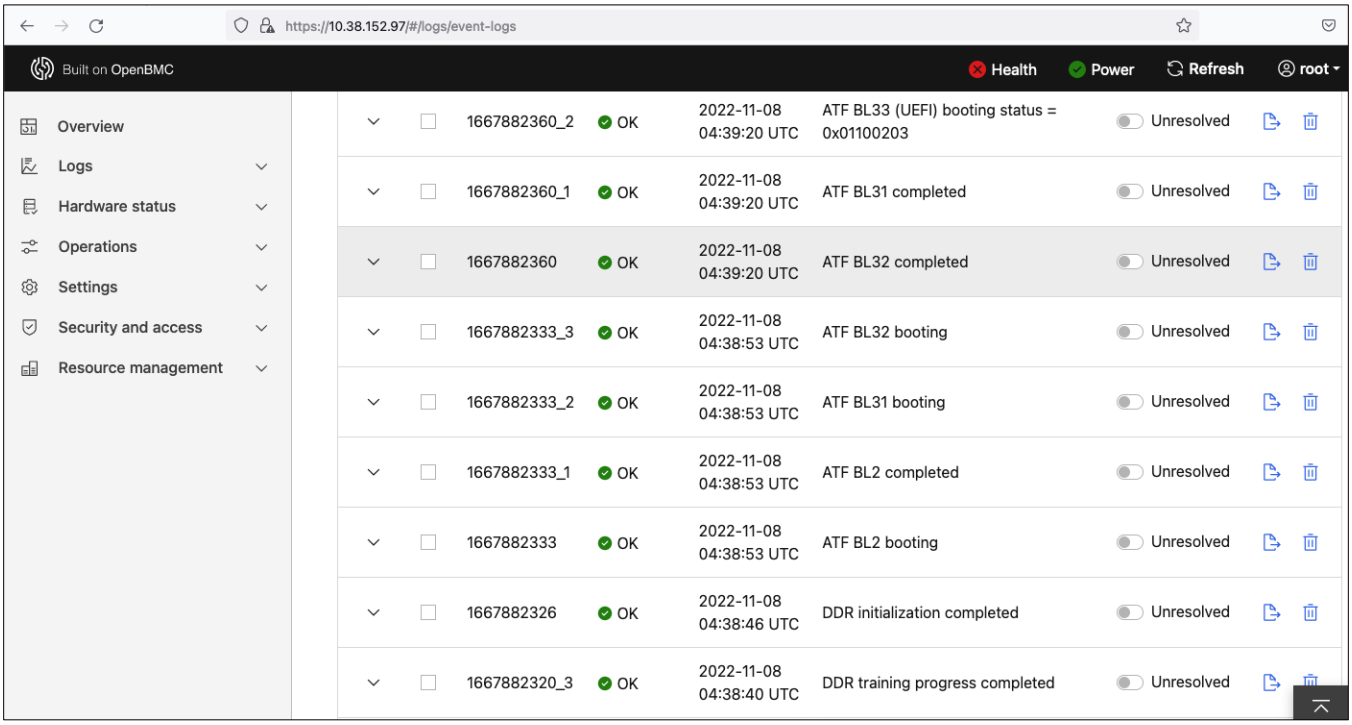
```
$ curl -X GET --user root:0penBmc -H "Content-Type: application/json" -H "If-match: " --
       insecure
       https://10.76.244.87/redfish/v1/Systems/system/LogServices/FaultLog/Entries/1/attachmen
       t --output bert.dump
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    56  100    56    0     0     62      0 --:--:-- --:--:-- --:--:--    62


$ hexdump bert.dump
0000000 0000 0000 0024 0000 8047 00d2 0001 0000
0000010 0801 00d5 9000 20d2 a015 20f2 0038 0000
0000020 801c 00d2 0002 0000 0014 0000 8000 40d2
0000030 0001 0000 0000 0000
0000038
```

Note: Upgrading BMC firmware clears FaultLog and Crashdump data.

6.4 PLDM Boot Progress

The PLDM sends boot progress information to the BMC using boot sensor events. The BMC gets boot sensor events from the PLDM, and logs all received events in the Redfish EventLog.

Figure 6: PLDM Boot Progress



Note: Boot progress information from the PLDM is not updated to the Redfish ComputerSystem's boot progress.

7. Serial Over LAN (SOL)

By default, all UARTs are configured to connect to the BMC.

The BMC supports the SOL feature, which enables users to connect to Host UARTs over BMC. By default, all console ports are configured to be directed to the BMC, and the output from these UARTs are logged.

While all platforms support CPU and ATF console ports, depending on platform hardware, other console ports like SCP (Altra-based platform, as in Mt.Jade), Mpro and SECPro (AmpereOne and later, as in Mt.Mitchell) might or might not be supported.

If supported, the information is as provided in the following table.

Table 9: UART Console Log Files

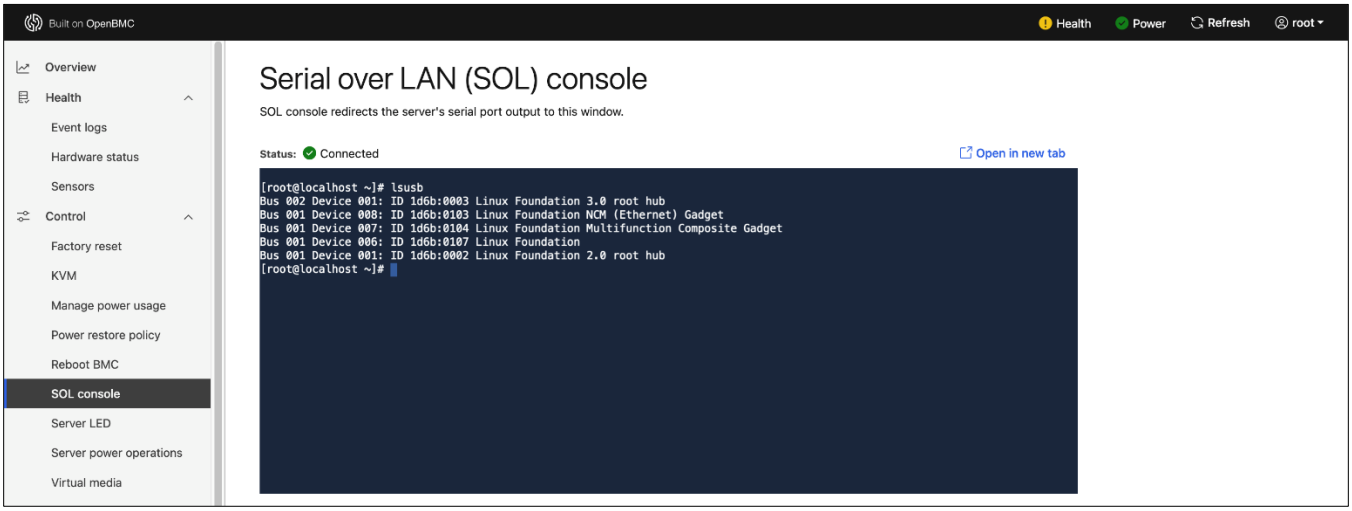
UART PORT	LOG FILE	FILE SIZE THRESHOLD	MAX FILE NUMBER
CPU	/var/log/obmc-console-cpu.log	256KB	2
ATF	/var/log/obmc-console-atf.log	64KB	2
Socket 0 Mpro	/var/log/obmc-console-mpro0.log	64KB	2
Socket 0 SECpro	/var/log/obmc-console-secpro0.log	64KB	2
Socket 1 Mpro	/var/log/obmc-console-mpro1.log	64KB	2
Socket 1 SECpro	/var/log/obmc-console-secpro1.log	64KB	2
Socket 0 SCP	/var/log/obmc-console-scp0.log	64KB	2
Socket 1 SCP	/var/log/obmc-console-scp1.log	64KB	2

Each log file is polled every minute to check if the file size exceeds the threshold value. When it exceeds the threshold value at polling time, the data in the log file is rotated to the next file <filename>.log.<file_num>, and a new empty file (with the same old file name) is created for the new console log. For example, when the size of **/var/log/obmc-console-cpu.log** is larger than 256 KB after one minute since the last polling, the data is copied to **/var/log/obmc-console-cpu.log.1** and a new log file **/var/log/obmc-console-cpu.log** is created with empty size (0 KB). The maximum file number is the number of log files that are allowed to be saved for one port at a time. When it reaches the maximum file number and new logs arrive, the oldest file is removed, and all the files are rotated to adopt new logs.

7.1 WebUI SOL

The BMC supports web-based SOL for the CPU UART. This is activated using the WebUI at **Control-> Serial over LAN console**.

Figure 7: SOL Console



7.2 IPMI SOL

IPMI SOL is supported only for the CPU UART with SOL payload enabled for all users and minimum privilege level is administrator.

An administrator can disable IPMI SOL access for a certain user using the standard IPMI command:

```
$ ipmitool sol payload disable <channelID> <userID>
```

For example: Disable user ID 2 in IPMI channel 2:

```
$ ipmitool sol payload disable 2 2
```

An administrator can also make some changes to IPMI SOL parameters (currently, only some of the commands are supported, there might be more in addition to this.) using standard out-of-band IPMI SOL set commands. For example:

- Change minimum privilege level to USER:

```
$ ipmitool -U root -P 0penBmc -C 17 -I lanplus -H BMC_IP sol set privilege-level user
```

To check current SOL parameters, use out-of-band IPMI SOL information command:

```
$ ipmitool -U root -P 0penBmc -C 17 -I lanplus -H BMC_IP sol info
Set in progress                 : set-complete
Enabled                         : true
Force Encryption                : false
Force Authentication            : false
Privilege Level                 : USER
Character Accumulate Level (ms) : 0
Character Send Threshold        : 1
Retry Count                     : 7
Retry Interval (ms)             : 1000
Volatile Bit Rate (kbps)        : IPMI-Over-Serial-Setting
Non-Volatile Bit Rate (kbps)    : IPMI-Over-Serial-Setting
Payload Channel                 : 2 (0x02)
Payload Port                    : 623
```

Note: Only the session owner and administrator can de-activate an IPMI SOL session.

7.3 SSH SOL

The BMC can connect to the console ports using SOL SSH using the following command:

```
$ ssh -p <port> <username>@${BMC_IP}
```

Where <port> = 2200 + <port index> as shown in the following table.

Table 10: SSH SOL

CONSOLE	PLATFORM	PORT INDEX
CPU	all	0
ATF	all	2
Socket 0 SCP	Mt.Jade	1
Socket 1 SCP	Mt.Jade	3
Socket 0 Mpro	Mt.Mitchell	1
Socket 0 Secpro	Mt.Mitchell	3
Socket 1 Mpro	Mt.Mitchell	4
Socket 1 Secpro	Mt.Mitchell	5

To disconnect solssh, type '~.'

Example: Connect to CPU console through SOL SSH with BMC IP 10.38.153.105:

```
$ ssh -p 2200 root@10.38.153.105
```

7.4 SOL MUX Control

To switch UARTs to a header, use the ampere_uartmux_ctrl.sh utility from the BMC Linux console with the following syntax:

```
$ ampere_uartmux_ctrl.sh <port index +1> <dir>
```

Where:

- <port index>: follow port index mapping as listed in *Table 10.*
- <dir>: 1: connect to header, 2: connect to BMC.

Example: Configure CPU UART to direct to DP9 port:

```
$ ampere_uartmux_ctrl.sh 1 1
```

Notes:

1. The BMC cannot record the console log when switching the UART MUX to header.
2. Switching the UART mux to redirect UARTs to console ports is not recommended. Only developers must perform this operation.

8. System Event Log (SEL)

Table 11 shows the System Event Log (SEL) log files.

Table 11: SEL Log Files

UART PORT	LOG FILE	FILE SIZE THRESHOLD	MAX FILE NUMBER
IPMI	/var/log/ipmi_sel	64KB	4
Redfish	/var/log/redfish	64KB	4
WebUI	/var/log/redfish	64KB	4

Each log file is polled every minute to check if the file size exceeds the threshold value. When it exceeds the threshold value at polling time, the data in the log file is rotated to the next file <filename>.log.<file_num>, and a new empty file (with the same old file name) gets created for the new SEL log. For example, when the size of **/var/log/ipmi_sel** is larger than 64 KB after one minute since the last polling, the data is copied to **/var/log/ipmi_sel.1** and a new log file **/var/log/ipmi_sel** is created with empty size (0 KB). The maximum file number is the number of log files that are allowed to be saved for one log source at a time. When it reaches the maximum file number and new logs arrive, the oldest file is removed, and all the files are rotated to adopt new logs.

Logs shown using commands or in WebUI appear in Last-In First-Out (LIFO) order.

The following commands show how to achieve logs from different sources.

IPMI:

```
$ ipmitool -H <BMC_IP> -U root -P 0penBmc -I lanplus sel elist
```

Redfish Event Logs:

```
$ curl -X GET --user root:0penBmc -H "Content-Type: application/json" --insecure
       https://<BMC_IP>/redfish/v1/Systems/system/LogServices/EventLog/Entries
```

Redfish Fault Logs:

```
$ curl -X GET --user root:0penBmc -H "Content-Type: application/json" --insecure
      https://<BMC_IP>/redfish/v1/Managers/bmc/LogServices/FaultLog/Entries
```

WebUI:

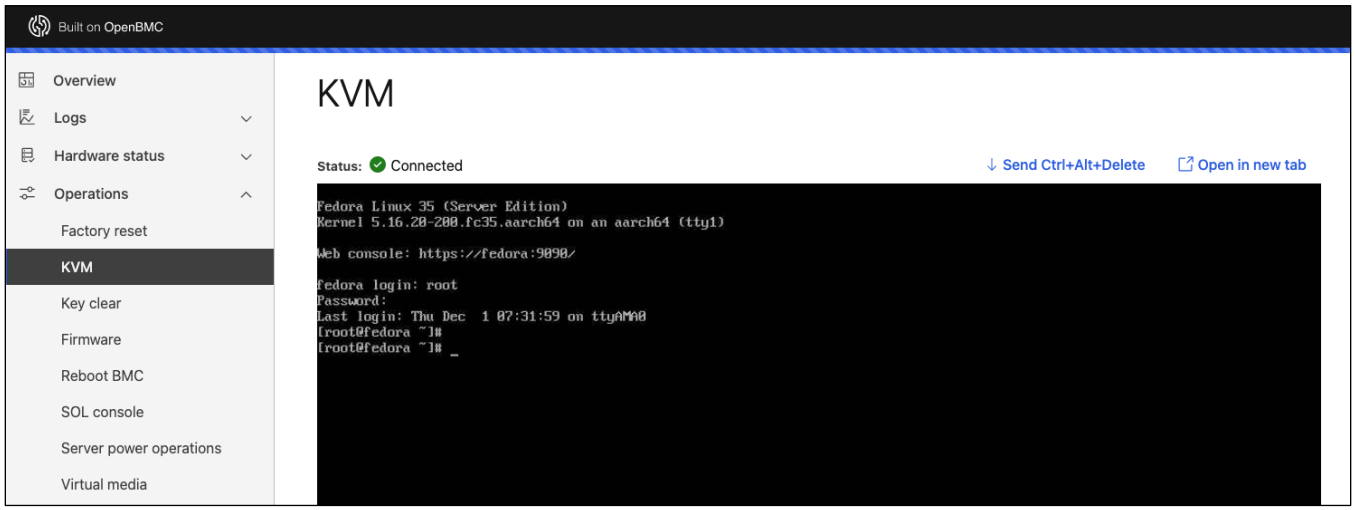
Log in to **WebUI -> Logs -> Event Logs**.

9. rKVM and vMedia

9.1 rKVM

The BMC supports KVM over WebUI. It is accessed using the WebUI at **Operations -> KVM**.

Figure 8: KVM



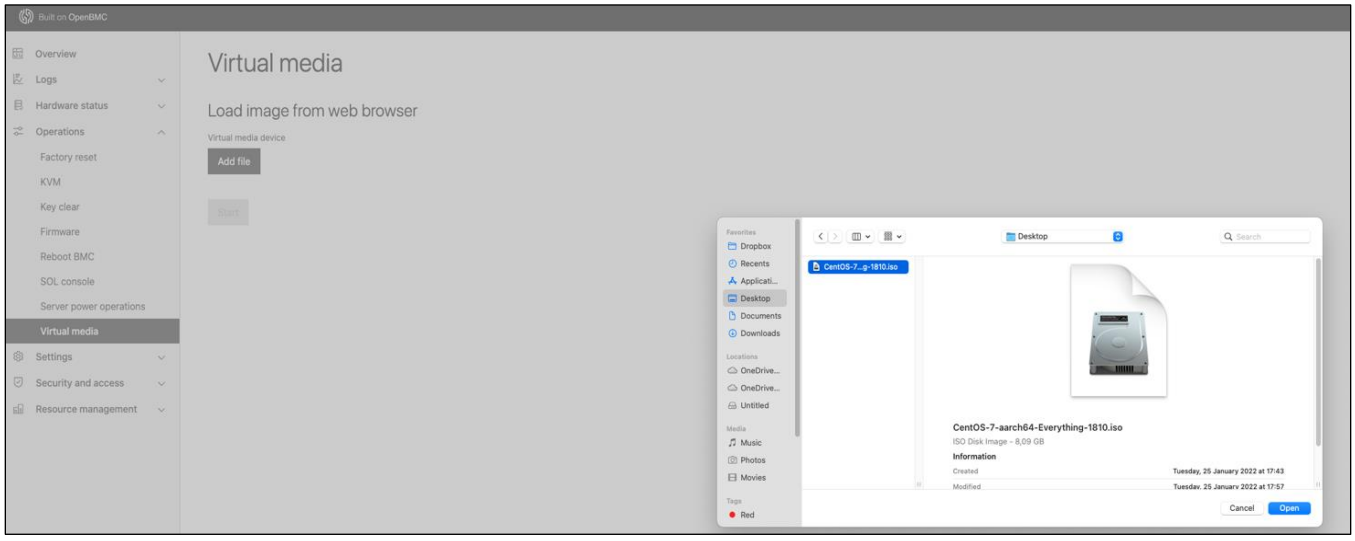
9.2 vMedia

9.2.1 VirtualMedia Support in WebUI

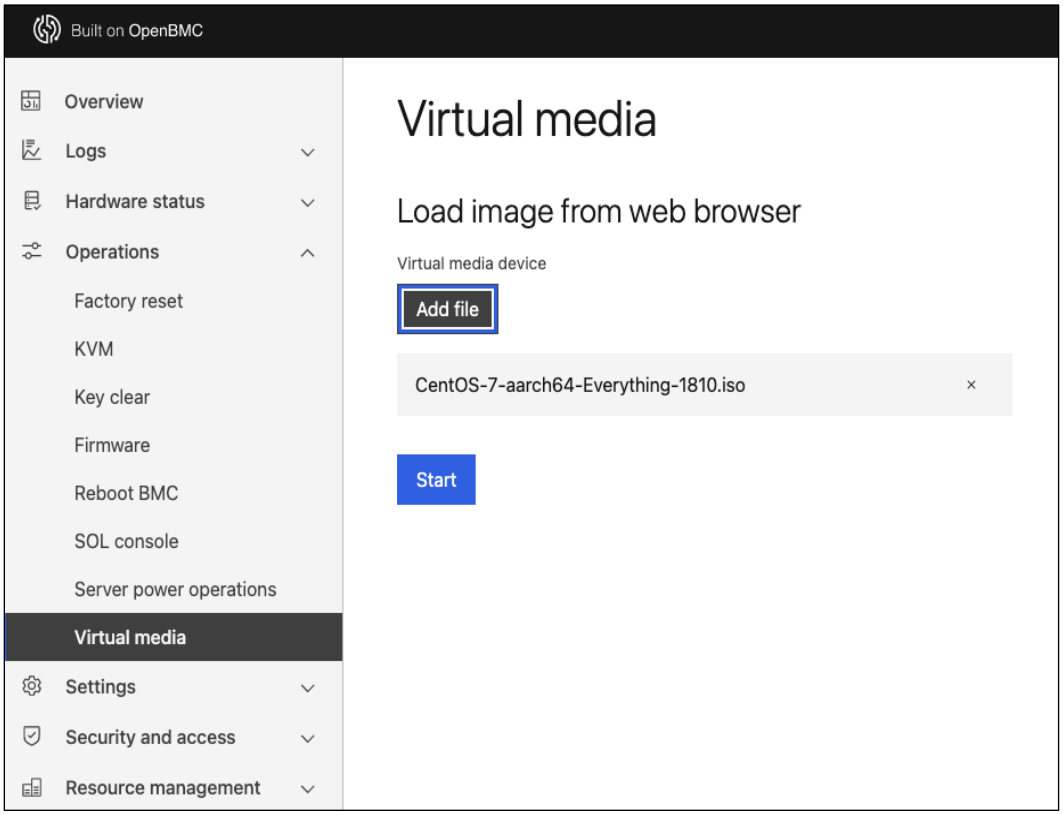
The BMC supports mounting of ISO files from the local machine (local media) over the WebUI.

1. Login to the WebUI.
2. Select **Operations -> Virtual media.**

Figure 9: VirtualMedia Support in WebUI



3. Click **Add file** and select the local ISO file.
4. Click **Open**.
5. Click **Start** to mount the ISO file.



Note: As the Virtual media connection is configured through WebUI to mount the local ISO file, it gets disconnected when the web session expires.

9.2.2 Redfish VirtualMedia Schema

The BMC supports mounting of the ISO file on Samba service over VirtualMedia schema. The following two operations are supported:

- InsertMedia: Mount the ISO file from Samba server to BMC and share it with the Host through virtual CD-ROM.

```
$ export token=`curl -k -H "Content-Type: application/json" -X POST https://${bmc}/login -d
       '{"username" :  "root", "password" :  "0penBmc"}' | grep token | awk '{print $2;}' | tr
       -d '"'`
$ curl -k -H "X-Auth-Token: $token" -H "Content-Type: application/json" -X POST
       https://${bmc}/redfish/v1/Managers/bmc/VirtualMedia/Slot_2/Actions/VirtualMedia.InsertM
       edia -d '{"Image": "smb://10.38.152.22/sambashare/debian-11.5.0-arm64-
       netinst.iso","UserName" : "ampsw","Password" : "ampsw1234"}'
```

- EjectMedia: Unmount the ISO file.

```
$ curl -k -H "X-Auth-Token: $token" -H "Content-Type: application/json" -X POST
       https://${bmc}/redfish/v1/Managers/bmc/VirtualMedia/Slot_2/Actions/VirtualMedia.EjectMe
       dia -d '{}'
```

10. IPMI

10.1 IPMI Encryption

The BMC supports only Cipher Suite 17 (CS17).

The ipmitool uses CS3 by default, users must add `'-C 17'` to the `ipmitool` command to run IPMI commands.

For example:

```
$ ipmitool -C 17 mc info
$ ipmitool -U root -P 0penBmc -I lanplus -C 17 -H 10.38.66.10 mc info
```

10.2 IPMI SSIF

IPMI SSIF provides an in-band solution for users to send IPMI from the Host without providing a username and password. During the Host kernel boot, it sends the following IPMI commands to get information from the BMC before creating the /dev/ipmi0 device node for the IPMI SSIF to work. It is required that the BMC boots successfully and the ssifbridged service is running. Otherwise, no /dev/ipmi0 device node is created. If this happens, the users must reboot the Host to recover.

Example of executing IPMI commands over SSIF:

```
# ipmitool sensor list
```

10.3 IPMI Power Restore Policy

The default power restore policy is Always-On. This means that the Host is automatically powered on after the system is AC powered and the BMC boot finishes.

Users can configure the power policy using IPMI `ipmitool chassis policy <state>` commands.

For example, to change power policy to "always-off":

```
$ ipmitool -C 17 chassis policy always-off
```

10.4 Ampere IPMI OEM Commands

See https://github.com/ampere-openbmc/ampere-ipmi-oem/blob/ampere/README.md for list of Ampere IPMI OEM commands supported.

10.5 SBMR IPMI Commands

The BMC supports IPMI commands defined in the SMBR 2.0 Specification and Redfish Host Interface specification. These commands include:

- Get Manager Certificate Fingerprint Commands (NetFn 2Ch, Command 01h)
- Get Bootstrap Account Credentials (NetFn 2Ch, Command 02h)
- Send Boot Progress Code (NetFn 2Ch, Command 02h)
- Get Boot Progress Code (NetFn 2Ch, Command 03h)

These IPMI commands are available only with the in-band SSIF.

Note: SBMR related features are not available for Mt.Jade.

10.5.1 Get Manager Certificate Fingerprint Commands

This IPMI command is to get SHA-256 SSL fingerprint data (32 bytes) from the BMC Web. The first two bytes are 52h (Group Extension identification) and 01h (SHA-256), and the next 32 bytes are fingerprint data.

```
# ipmitool raw 0x2c 0x1 0x52 0x1
52 01 bb 8d f1 ce 12 20 4e e9 ef b8 a6 99 08 85
6c 46 1d 18 2f f1 1d 87 27 ab 59 47 04 63 93 db
39 b4
```

Users can verify the fingerprint number using the following command:

```
$ echo | openssl s_client -connect 10.38.153.252:443 |& openssl x509 -sha256 -fingerprint -
       noout
SHA256
       Fingerprint=BB:8D:F1:CE:12:20:4E:E9:EF:B8:A6:99:08:85:6C:46:1D:18:2F:F1:1D:87:27:AB:59:
       47:04:63:93:DB:39:B4
```

10.5.2 Get Bootstrap Account Credentials

This command, when the CredentialBootstrapping Enabled property is true, is used to create a new bootstrap account to send Redfish commands from the Host.

10.5.3 Send Boot Progress Code

This IPMI command is specified in the SBMR 2.0 Specification. After receiving the command, the BMC performs the following:

- Stores the command information to its volatile memory. The command information is lost when the BMC is rebooted.
- Updates Redfish "Boot Progress" information.

```
{
    "BootProgress": {
        "LastState": "OEM",
        "OemLastState": "0x00112233445566778899" <-- 9 bytes of boot progress code
    },
}
```

The BMC also stores nine bytes of boot progress code to PostCodes.

```
$ curl -X GET --user root:0penBmc -H "Content-Type: application/json" \
    --insecure https://${BMC_IP}/redfish/v1/Systems/system/LogServices/PostCodes/Entries
{
  "@odata.id": "/redfish/v1/Systems/system/LogServices/PostCodes/Entries",
  "@odata.type": "#LogEntryCollection.LogEntryCollection",
  "Description": "Collection of POST Code Log Entries",
  "Members": [
    {
```

```
      "@odata.id": "/redfish/v1/Systems/system/LogServices/PostCodes/Entries/B1-1",
      "@odata.type": "#LogEntry.v1_9_0.LogEntry",
      "AdditionalDataURI": "/redfish/v1/Systems/system/LogServices/PostCodes/Entries/B1-
       1/attachment",
      "Created": "2022-11-15T23:15:32+00:00",
      "EntryType": "Event",
      "Id": "B1-1",
      "Message": "Boot Count: 1; Time Stamp Offset: 0.0000 seconds; POST Code:
       0x100000001100203",
      "MessageArgs": [
        "1",
        "0.0000",
        "0x100000001100203"
      ],
      "MessageId": "OpenBMC.0.2.BIOSPOSTCode",
      "Name": "POST Code Log Entry",
      "Severity": "OK"
    },
```


Note: The BMC does not keep the Boot Progress Code when it is rebooted.

10.5.4 Get Boot Progress Code

This IPMI command returns the 9-byte boot progress code sent most recently by the "Send Boot Progress Code" command.

11. FRU

11.1 FRU Reporting

The `ipmitool fru print` command is supported to display the FRU contents for the mainboard FRU EEPROM and various pluggable boards that contain FRU EEPROMs. And FRU ID for the motherboard is always zero. FRU ID for each platform follows the platform definition.

FRU ID definition for Mt.Mitchell platform is defined in *Table 12.*

Table 12: Mt. Mitchell FRU ID Definition

FRU ID	DESCRIPTION
0	FRU of the system, containing Manufacturer, Product, and Board information.
1	FRU of BMC board, containing Board information.
2	FRU of Front backplane 1, containing Board information.
3	FRU of Front backplane 2, containing Board information.
4	FRU of Front backplane 3, containing Board information.

Note: In Mt.Mitchell platform, when the Host is switched off, no backplane FRU is available.

Some examples for IPMI FRU report are listed as follows:

```
# ipmitool fru print 0
FRU Device Description : Builtin FRU Device (ID 0)
 Chassis Type          : Rack Mount Chassis
 Chassis Part Number   : ChassisPN
 Chassis Serial        : 222116639
 Board Mfg Date        : Thu May 18 03:15:00 2000 UTC
 Board Mfg             : Inspur
 Board Product         : MB
 Board Serial          : BoardSN
 Board Part Number     : BoardPN
 Product Manufacturer  : Inspur
 Product Name          : BoardPN
 Product Part Number   : ProductPN
 Product Version       : AmpereOne
 Product Serial        : 000MTC007
 Product Asset Tag     : NULL

# ipmitool fru print 1
FRU Device Description : BMC (ID 1)
 Board Mfg Date        : Fri Feb 11 00:00:00 2022 UTC
 Board Mfg             : Inspur
 Board Product         : Mitchell BMC DC-SCM
 Board Serial          : CAN304GL0084X02
 Board Part Number     : YZCA-03092-101
 Board Extra           : B4:05:5D:E1:01:84
```

11.2 BMC MAC Address Storage

The MAC address for the BMC Reduced Gigabit Media-Independent Interface (RGMII) port is stored in the BMC FRU EEPROM device at **Board Information Area > Customer Field 1**. When the BMC finishes booting, the BMC reads the MAC address stored in the FRU EEPROM and uses it for the BMC RGMII port (eth0 interface).

Note: As the BMC MAC address is allocated by the BMC manufacturer, the users are not expected to change it.

11.3 System UUID

The system UUID is stored in the FRU EEPROM in the multi-record section. The Universally Unique Identifier (UUID) is read and reported in Redfish under **/redfish/v1/Systems/system**.

```
$ curl -k --user root:0penBmc --insecure -X GET https://${bmc}/redfish/v1/Systems/system |
      grep UUID
  "UUID": " 44f57056-b304-11eb-b57c-0cc47ad8d1fc"
```

The IPMI command `mc guid` reports the Globally Unique ID (GUID):

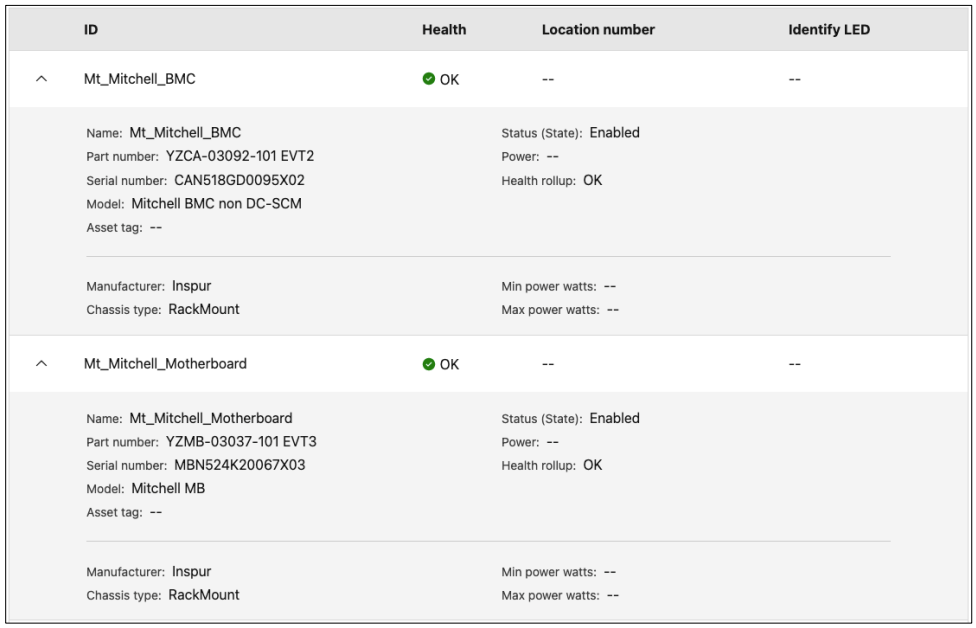
```
root@mtmitchell:~# ipmitool mc guid
System GUID   : 44f57056-b304-11eb-b57c-0cc47ad8d1fc
GUID Encoding : IPMI
GUID Version  : Time-based
Timestamp     : 05/12/21 09:27:28 UTC
```

If the mainboard FRU EEPROM does not contain the GUID, the BMC randomly generates a UUID value to use with Redfish and IPMI.

11.4 Inventory Data

To easily differentiate between platform systems, Board information (including Manufacturer, Model, Part Number, and Serial Number) in the FRU is used as inventory data for chassis instances.

Figure 10: Inventory Data



Note: The inventory information is mapped only when the BMC boots. Changing the FRU contents will take effect only after the next BMC boot.

12. Fan Control

12.1 Default Fan Speed

The fan speed is set to 60% duty cycle by default.

After starting, the automatic fan control service uses its algorithm to control fan speeds. Before the automatic fan control service stops, it returns fan speeds back to 60% duty cycle.

12.2 Manual Fan Control

To manually control the fans, the fan control service must be disabled. In these examples, <fan> is the fan index corresponding to the label on the chassis (valid fan indexes are 0 to 5) and <duty> is the fan duty cycle, which ranges from 1 to 100.

The following commands get and set the status of the fan control service, where 1 is disabled and 0 is enabled.

1. To get the fan control service status:

```
# ampere_fanctrl.sh getstatus
```

2. To set fan control service status:

```
# ampere_fanctrl.sh setstatus <0|1>
```

These commands set the fan speed and check the fan speeds for fans on the system.

1. To set fan speed for a fan:

```
# ampere_fanctrl.sh setspeed <fan> <duty>
```

2. To report fan speed for fans:

```
# ampere_fanctrl.sh getspeed <fan>
```

12.3 Automatic Fan Control

After the host is powered on and the PLDM sensor is available, fans are controlled automatically using a current fan control table.

Fans are controlled based on the sensor values. If the sensor value reading is outside of the hysteresis bounds, use the reading to control the fans according to the fan control table. Otherwise, they use the last reading:

1. When the sensor value is increasing, |currentValue – lastValue| > Negative Hysteresis is the condition to update the fan speeds and lastValue.
2. When the sensor value is decreasing,|currentValue – lastValue| > Positive Hysteresis is the condition to update the fan speeds and lastValue.

13. Chassis

13.1 Buttons

13.1.1 Power Button

The power button can be used to control the Host power as follows:

- When the Host is OFF, pressing the power button executes a DC power ON (both short press and long press).
- When the Host is ON, pressing the power button for five seconds (or longer) turns the DC power OFF. Pressing the power button for less than five seconds gracefully shuts down the Host.

13.1.2 Reset Button

The reset button can be used to reset the Host:

- When the Host is ON, pressing the reset button gracefully shuts down the host, and turns the DC power OFF and then DC power ON.
- When the Host is OFF, pressing the reset button does not have any effect.

13.1.3 UID Button

The UID (or identify) button can be used to turn ON and OFF the UID (identify) LED:

- When the UID LED is ON, pressing the UID button turns the UID LED OFF.
- When the UID LED is OFF, pressing the UID button turns the UID LED ON.
- When the UID button is Blinking, pressing the UID button does not have any effect.

13.2 LEDs

The BMC supports the LED functions summarized in *Table 13*.

Table 13: LED Indications

LED	STATE	COLOR	DESCRIPTION
UID LED	On	Blue	The system is identified remotely using the IPMI chassis identify command or when the UID button is pushed from the Off state.
	Blinking	Blue	Set by patching to the IndicatorLed property to "Blinking" on the Redfish ComputerSystem schema. The blinking status is only changed to Off when a command of another patching with IndicatorLed property "Off" is sent to the Redfish schema.
	Off	None	–
Fault LED (System Status LED)	On	Red	Indicates critical failures occurred, including the following events: • Fan failure (refer to Fan Status LED) • Over-temperature. (This LED is turned on when this event occurs and automatically turned off after approximately 10 seconds.) • PSU failure (refer to the Power Status LED) • Power Fault • Uncorrectable errors. (This LED is on until the host is rebooted.) The BMC turns off the Fault LED only when all critical failures are cleared.

LED	STATE	COLOR	DESCRIPTION
	Off	None	No failure.
Fan Status LED	On	Red	The BMC turns on the Fan Status LED when the following failures occur in approximately 25 seconds: • The fan is unplugged. • A fan is running under the expected speed (less than 500 RPM). The BMC turns off the Fan Status LED only when all fan failures are cleared. **Note:** The Fan Status LED requires the Fan Control Service to work. When the Fan Control Service is disabled, the Fan Status LED is no longer updated.
	Off	None	No failure.
Power Status LED	On	Red	A PSU failure occurred. The BMC monitors the PSUs using the PMBus and turns on the Power Status LED for the following failures: • One of the power cores is missing. • One of the bits in the STATUS_BYTE register, which is defined in the PMBus Specification, is set to 1 (including under-voltage faults, over-voltage faults, and over-current faults). The BMC turns off the Power Status LED only when all PSU failures are cleared.
	Off	None	No failure.

Notes:

- Disabling the Fan speed control disables the Fan failure check, so there are no Fan Status LED and Fault LED changes.
- Not all platforms have all the mentioned LEDs. If a platform misses any LED from the mentioned LEDs, related functions are not available.

13.3 Chassis Intrusion

Chassis intrusion is supported to detect if the chassis cover is opened unexpectedly. Users can get the chassis intrusion status using IPMI and Redfish.

1. To get chassis intrusion status using IPMI:

```
# ipmitool chassis status | grep Intrusion
Chassis Intrusion: inactive
```

   where "inactive" means not intruded and "active" means intruded.

2. To get chassis intrusion status using Redfish:

```
# curl -X GET --user root:0penBmc -H "Content-Type: application/json" \
       --insecure https://<BMC_IP>/redfish/v1/Chassis/chassis/
{
…
"PhysicalSecurity": {
    "IntrusionSensor": "Normal",
    "IntrusionSensorNumber": 1,
```

```
      "IntrusionSensorReArm": "Manual"
    },
…
}
```

The following two modes are supported:

1. Automatic:
   a. When the chassis is opened, the intrusion status changes to HardwareIntrusion.
   b. When the chassis is closed, the intrusion status changes to Normal.
2. Manual: Once the chassis is opened, the intrusion status changes to HardwareIntrusion and keeps this state even when the chassis covered is then closed. Users need to change the value to Normal using Redfish.

```
# curl -X PATCH --user root:0penBmc -H "Content-Type: application/json" \
      -H "If-match:*" --insecure \
https://<BMC_IP>/redfish/v1/Chassis/chassis/
  -d \
      '{ "PhysicalSecurity": { "IntrusionSensor": "Normal" } }'
```

The default mode is Manual.

When users set the intrusion status to Normal, the chassis opens. The intrusion status is changed to Normal and the same value is maintained until the chassis is closed and opened again.

Note: Chassis intrusion is not available for Mt.Jade.

14. Host Supported Features

14.1 Firmware Hang Detection

The firmware hang detection feature monitors the Socket 0 Mpro heartbeat GPIO signals after the SYS_PWRGD signal is asserted to detect if the Mpro hangs. When this occurs, the BMC automatically reboots the Host.

To disable this feature, create a file named **/var/ampere/sysfw-hang-disable**.

Note: This feature is not available in Mt.Jade.

14.2 Host Secure Provisioning

The BMC supports flashing the special image and asserting the SPECIAL_BOOT GPIO pin and rebooting the Host to boot the host into secure mode, then de-assert the pin when the Host completes processing and asserts FW_BOOT_OK signal.

Steps to execute:

- Login to the BMC console using root/0penBmc
- Copy the special image to the /tmp folder. For example:

```
# scp ampsw@10.38.153.24:~/mtmitchell/spinor_special.img .
```

- Run the ampere_flash_bios.sh script to flash the image to the main Host SPI-NOR and start the secure provisioning process:

```
# ampere_flash_bios.sh <secure image.img> 1 1
```

The following message might appear from the host console:

```
[00:00:00.248,000] <inf> scu_stat: is_special_boot() = 1
[00:00:00.383,000] <inf> SECpro_cert_ext: Assert FW_BOOT_OK
[00:00:00.383,000] <inf> SECpro_cert_ext: Deassert FAULT_ALERT
[00:00:00.383,000] <inf> SECpro_cert_ext: SEC provision completed successfully
```

- Turn OFF the Host and flash back to normal Host firmware image.

When the Host boots with SPECIAL_BOOT asserted, the following actions take place.

- The Host boots with special image flashed above.
- SECProv authenticates and processes the special certificate:
  - If fail - Assert FW_BOOT_OK pin and SYS_AUTH_FAIL pin. Pulse GPIO fault code over GPIO_FAULT pin and loop forever
- SECProv blows fuses per the specific operation indicated in the special certificate.
- SECProv reads back fuses to verify fuse blowing.
- SECProv reports success/fail to the BMC:
  - If success - assert FW_BOOT_OK and print a success message over the UART.
  - If fail – assert FW_BOOT_OK and SYS_AUTH_FAIL pin. Pulse GPIO fault code over GPIO_FAULT pin and loop forever.

14.3 Host Firmware Revision

The OpenBMC supports host firmware revision for running and backup images:

- Running image: The host firmware revision for the running host firmware.
- Backup image: Temporary host firmware revision that is to be flashed to Host SPI-NOR. This is cleared after the completion of the host firmware update.

The running image host firmware revision is set as the following:

- By default, Host firmware revision is not set.

- When users run the Host firmware update, either to the main or secondary Host SPI-NOR:
    - The firmware revision from the MANIFEST file (version field) is set to the backup image's host firmware revision.
    - After completion of the Host firmware update, the firmware revision above is set to the running image, and the one from the backup image is cleared.

    Note: Host firmware from firmware update (specified in the MANIFEST file) is not backed up, so it is lost when the BMC reboots.
- During the Host UEFI firmware boots, the Host firmware revision is set to BMC using the Ampere IPMI OEM "Set Host Firmware Revision" command (0x3c 0xf0). The Host firmware through this command is stored in BMC non-volatile memory and is recovered after the BMC reboots.
- Before the Host UEFI boots OS, it sends SMBIOS data to the BMC through IPMI blob commands. Then the BMC parses SMBIOS data to get the firmware revision from SMBIOS Type 0's Bios Version field to update the Host firmware revision, overwritten to previous value.

To get Host firmware revision for the running image:

```
$ curl --user root:0penBmc --insecure -H "Content-Type: application/json" -X GET \
    https://<BMC_IP>/redfish/v1/UpdateService/FirmwareInventory/bios_active \
    | grep Version
  "Version": "03.03.00003001"
```

Note: SMBIOS data is erased in BMC factory reset.

14.4 NMI Trigger

The BMC supports NMI trigger through Redfish ComputerSystem schema, shown as follows:

```
$ curl -k -H "X-Auth-Token: $token" -H "Content-Type: \
   application/json" -X POST \
   https://<BMC_IP>/redfish/v1/Systems/system/Actions/ComputerSystem.Reset \
    -d '{"ResetType": "Nmi"}'
```

The BMC also supports NMI trigger through a standard IPMI command, shown as follows:

```
$ ipmitool chassis power diag
```

Note: NMI trigger is not supported for Mt.Jade.

14.5 Scandump Mode Support

Note:

1. Scandump mode is not supported for Mt.Jade.
2. On Mt.Mitchell platform, this feature requires CPLD 0.75 and above.

The BMC provides Ampere IPMI OEM commands Get/Set Scandump mode (0x3c 0x25/0x26) to enter/exit "Scandump mode".

See the following documents for more information.

- *AmpereOne IPMI OEM Command Specification* for command description.
- *AmpereOne BMC Software Functional Specification* for the BMC behavior in Scandump mode.

The following is the instruction flow for Scandump mode:

1. Enable Scandump mode
2. After Scandump mode is complete, power off the Host
3. Disable Scandump mode
4. Power on the Host

Invalid user actions when Scandump mode is active:

1. Reboot BMC

- As CPLD masks all CPU GPIOs, including SYS_PWRGD, in the scan dump mode, the BMC fails to detect current Host status and the incorrect CPU status is reported and CPU related sensors are not loaded.
- For recovery, power off the Host and then disable Scandump mode before powering on the Host again.

2. Power control (cycle/reset) Host

- CPLD masks all CPU GPIOs and SYS_PWRGD, in Scandump mode. When users power cycle or reset the Host, as no SYS_PWRGD signal is asserted, the BMC incorrectly interprets the Host's failure to boot. Consequently, CPU related sensors also fail to be enabled.
- To recover, power off the Host and then disable Scandump mode before powering on the Host again.

14.6 DCMI Power Limit OEM Action

The BMC supports DCMI power limit action OEM_02 to run Ampere defined actions. See *AmpereOne BMC Software Functional Specification* for details on what the action supports.

To configure the action:

```
$ ipmitool dcmi power set_limit action oem_02
```

Other configurations such as set power limit, correction time, and so on, follow IPMI DCMI commands.

15. Network

15.1 Network Connectivity Status Indicator (NCSI)

Ampere OpenBMC supports NCSI for OOB IPMI, Redfish and WebUI access. However, no failover capability is supported.

Depend on platform design and BMC CPU (lile Aspeed AST2500, AST2600, and so on) used, it is different in the slot and Ethernet interface between supported platforms:

- Ethernet interface for NCSI in Mt.Jade is eth0 while the interface for the RGMII port is eth1.
- Mt.Mitchell has two OCP slots on the motherboard but just supports NCSI for the OCP card plugged on slot 0. And the Ethernet interface for NCSI is eth1.

16. Engineering Tools

16.1 nvparm

The nvparm tool is an engineering tool running on the BMC Linux console. The nvparm tool enables users to edit fields in Non-Volatile Parameters (NVPARAM).

There are two different nvparm tools, both generate nvparm utility:

- ac01 nvparm, located at https://github.com/ampere-openbmc/ampere-misc/tree/ampere/subprojects/ac01-nvparm, is used for Altra-based platform like Mt.Jade.
- NVP Management Tool, located at https://github.com/AmpereComputing/NVP-Management-Tool, is used for platforms that use AmpereOne and later Host CPUs, like Mt.Mitchell. See https://github.com/AmpereComputing/NVP-Management-Tool/blob/main/README.md for all supported command options.

On the Ampere reference platforms, a GPIO MUX must be controlled to open the SPI bus from the BMC to the Host SPI-NOR or the bootstrap EEPROM. Perform the following steps to access the NVPARAM from the BMC console:

1. Log in to the BMC console with the root account.
2. Power off the host using ipmitool.

```
# ipmitool chassis power off
```

3. Bind the Host SPI-NOR device.

```
# echo spi1.0 > /sys/bus/spi/drivers/spi-nor/bind
```

4. Execute the nvparm tool to modify NVPARAM fields. For example, in Mt.Mitchell BMC console:

```
# nvparm -t nvpd -f nvpdddr0.nvp -i 0 -w 0xffffffffffffffff -v 0x01
```

5. Unbind the Host SPI-NOR device.

```
# echo spi1.0 > /sys/bus/spi/drivers/spi-nor/unbind
```

Note: The host SPI-NOR cannot bind if it is already bound. The BMC automatically switches GPIOs when turning the host off and on so that the users do not have to do it manually.

16.2 Common utilities

16.2.1 GPIO utilities

16.2.1.1 gpioset

The gpioset utility is used to set value for an output GPIO. It configures the specified GPIO as output then sets the new value for the GPIO.

16.2.1.2 gpioget

The gpioget utility is used to get value for an input GPIO. It configures the specified GPIO as an input pin before reading the value.

17. References

- *Platform Management FRU Information Storage Definition, v1.0,* February 28, 2013, Intel Corporation.
- *IPMI Interface Specification Second Generation 2.0, Revision 1.1,* October 1, 2013, Intel Corporation.
- DEN0069C Arm® Server Base Manageability Requirements (SBMR) 1.1 – Platform Design Document.
- *AmpereOne BMC Software Functional Specification*

18. Revision History

ISSUE	DATE	DESCRIPTION
0.90	April 19, 2024	Updated: • *Section 1.1, Scope* • *Section 1.5, Firmware download* • *Section 3.1, BMC Firmware Updates* • *Section 3.1.3.1, Prepare the BMC Firmware Package for Upgrade* • *Section 3.2, Host and Device Firmware Updates* • *Section 3.2.2.1, Host Firmware Package* • *Section 4.1, Control Power using IPMI, WebUI, and Redfish* • *Section 4.2, Control Power from Host OS* • *Section 5.3, Sensor Monitoring using Redfish* • *Section 6, Platform Level Data Model (PLDM)* • *Section 7, Serial Over LAN (SOL)* • *Section 7.3, SSH SOL* • *Section 7.4, SOL MUX Control* • *Section 10.4 , Ampere IPMI OEM Commands* • *Section 10.5, SBMR IPMI Commands* • *Section 11, FRU* • *Section 11.4, Inventory Data* • *Section 13.2, LEDs* • *Section 13.3, Chassis Intrusion* • *Section 14.1, Firmware Hang Detection* • *Section 14.2, Host Secure Provisioning* • *Section 14.3, Host Firmware Revision* • *Section 14.4, NMI Trigger* • *Section 14.5, Scandump Mode Support* • *Section 15.1, Network Connectivity Status Indicator (NCSI)* • *Section 16.1, nvparm* • *Section 16.2, Common utilities* • *Section 17, References*
0.85	September 15, 2023	Updated: • Section 1.4.2, Revision Reports • Section 5, Sensor Monitoring • Section 5.1, Sensor Monitoring using IPMI • Section 5.3, Sensor Monitoring using Redfish • Section 7.2, IPMI SOL • Section 8, System Event Log (SEL) • Section 9.2.2, Redfish VirtualMedia Schema • Section 11.1, FRU Reporting • Section 14.5, Scandump Mode Support
0.80	May 8, 2023	Added: • Section 2.3, Ampere Password Policy • Section 14.5, Scandump Mode Support Updated: • Section 1.1, Scope • Section 1.3.2, Revision Reports

ISSUE	DATE	DESCRIPTION
		• Section 3.1.3.1, Prepare the BMC Firmware Package for Upgrade • Section 3.1.3.2, Update BMC Firmware using Redfish • Section 3.2.2.1, Host Firmware Package • Section 5.2.1, Sensor Monitoring using IPMI • Section 5.2.3, Sensor Monitoring using Redfish • Section 10.4, Ampere IPMI OEM Commands • Section 11.1, FRU Reporting • Section 16.1.1, Supported Options • Section 16.1.2, Accessing NVPARAM from the BMC Console (deleted the previous section *16.1.2* titled *Commands Containing GUIDs*) Deleted: • Section 14.2, JTAG TRST Workaround
0.75	March 27, 2023	*Updated:* • *Section 1.3.2, Revision Reports* • *Section 2.2, Account Policy Settings* • *Section 3.1.1, Recover BMC Firmware from U-boot* • *Table 3: Purpose and ExtendedVersion in MANIFEST File* • *Section 5.1, Sensor Data* • *Section 6.2, Redfish RAS Error Report* • *Section 6.3, BERT Crash Dump Report* • *Section 7, Serial Over LAN (SOL)* • *Section 13.1.3, UID Button* • *Section 14., Host Firmware Revision* • *Section 14.4, NMI Trigger* *Added:* • *Section 8, System Event Log (SEL)* • Minor fixes and corrections

ISSUE	DATE	DESCRIPTION
0.70	February 15, 2023	Updated: • Section 1.12.2, Scope. • Section 3.1.3.1, Prepare the BMC Firmware Package for Upgrade. • Section 3.2.1, Update Host and Device Firmware using the BMC Console • Table 4: Power Control from IPMI, WebUI, and Redfish • Table 5: Power Control from Host OS • Section 6.2, Redfish RAS Error Report • Section 7, Serial Over LAN (SOL) • Section 10.4, Ampere IPMI OEM Commands • Section 13.1.2, Reset Button • Section 14.1, Firmware Hang Detection • Section 16.1., Accessing NVPARAM from the BMC Console Added: • Section 6.3, BERT Crash Dump Report • Table 7: UART Console Log Files • Section 9, rKVM and vMedia and subsections. • Section 14.2, Host Secure Provisioning • Section 14.3, Host Firmware Revision • Section 14.4, NMI Trigger Minor fixes and corrections
0.65	December 1, 2022	Updated: • Section 2.2, Account Policy Settings. • Section 7.2, IPMI SOL. • Section 7.3, SSH SOL. • Section 11.1, FRU Reporting. • Section 12.3, Automatic Fan Control. Added: • A note in Section 3.1.3.1, Prepare the BMC Firmware Package for Upgrade. • Section 6, Platform Level Data Model (PLDM). • Section 10.2, IPMI SSIF • Section 10.5, SBMR IPMI Commands and subsections • Table 9 • Section 13, Chassis. • Section 14, Host Supported Features. • Section 15, Network. Deleted: • *Section 4.3*, *Power Buttons* and *Section 4.4 Rest Buttons*. Minor formatting edits.

ISSUE	DATE	DESCRIPTION
0.60	September 2, 2022	Updated: • Section 3.2.1, Update Host and Device Firmware using the BMC Console • Section 4.3, Power Control Restriction • Section 5.1, Sensor Data • Section 5.3, Limitations • Section 10.4, Ampere IPMI OEM Commands • Section 11.4, Inventory Data • Section 13.1.2, Reset Button • Section 16.1.1, Supported Options • Section 16.1.2, Commands Containing GUIDs Deleted: • *Server Based Manageability Requirements (SBMR) Support*.
0.50	July 11, 2022	Initial issue.

April 19, 2024

Ampere Computing reserves the right to change or discontinue this product without notice.

While the information contained herein is believed to be accurate, such information is preliminary, and should not be relied upon for accuracy or completeness, and no representations or warranties of accuracy or completeness are made.

The information contained in this document is subject to change or withdrawal at any time without notice and is being provided on an "AS IS" basis without warranty or indemnity of any kind, whether express or implied, including without limitation, the implied warranties of non-infringement, merchantability, or fitness for a particular purpose.

Any products, services, or programs discussed in this document are sold or licensed under Ampere Computing's standard terms and conditions, copies of which may be obtained from your local Ampere Computing representative. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Ampere Computing or third parties.

Without limiting the generality of the foregoing, any performance data contained in this document was determined in a specific or controlled environment and not submitted to any formal Ampere Computing test. Therefore, the results obtained in other operating environments may vary significantly. Under no circumstances will Ampere Computing be liable for any damages whatsoever arising out of or resulting from any use of the document or the information contained herein.

Ampere Computing
4655 Great America Parkway, Santa Clara, CA 95054
Phone: (669) 770-3700
https://www.amperecomputing.com

Ampere Computing reserves the right to make changes to its products, its datasheets, or related documentation, without notice and warrants its products solely pursuant to its terms and conditions of sale, only to substantially comply with the latest available datasheet.

Ampere, Ampere Computing, the Ampere Computing and 'A' logos, and Altra are registered trademarks of Ampere Computing.

Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All other trademarks are the property of their respective holders.