

Engineering Guidelines Document

Document No. SW-EGD-2021-OCP-UGD

*AMI MegaRAC® OCP OpenBMC*

*TiogaPass User Guide*

Revision: 0.1

*Published by Software R&D*

AMI-CONFIDENTIAL

Copyright ©American Megatrends, Inc.

3095 Satellite Boulevard, Building 800 Suite 425,

Duluth,

Georgia 30096,

U.S.A.

All Rights Reserved

Confidential, Unpublished Document

Property of American Megatrends, Inc.

This page is intentionally left blank

Document Change History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Doc Version** | **Author** | **Description** |
| 04-04-2021 | 0.1 | Hongwei Zhang | Initial Draft |
| 04-13-2021 | 0.2 | Hongwei Zhang | Add SPI, Network Interface |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[1. Introduction 4](#_Toc69214921)

[2. Supported Features in OpenBMC 5](#_Toc69214922)

[2.1. BMC Hardware Component 5](#_Toc69214923)

[2.2. BMC UART Console Access 5](#_Toc69214924)

[2.3. BMC SPI Flash 6](#_Toc69214925)

[2.4. Network Interfaces 6](#_Toc69214926)

[2.5. LAN Interface 6](#_Toc69214927)

[2.6. KCS Interface 7](#_Toc69214928)

[2.7. Sideband Interface 7](#_Toc69214929)

[2.8. Sensor Support 7](#_Toc69214930)

[2.9. System Event Log Support 8](#_Toc69214931)

[2.10. FRU Device Support 9](#_Toc69214932)

[2.11. BMC Firmware Upgrade Support 10](#_Toc69214933)

[2.12. WebUI Support 10](#_Toc69214934)

[2.13. KVM Support 13](#_Toc69214935)

[2.14. Reboot BMC 14](#_Toc69214936)

[2.15. Server Power Operation 15](#_Toc69214937)

[2.16. NTP Support 16](#_Toc69214938)

[2.17. BMC Network Settings 17](#_Toc69214939)

[2.18. User Management Support 18](#_Toc69214940)

[2.19. IPMI 2.0 and DCMI 1.5 support: 19](#_Toc69214941)

[2.20. vMedia Support 19](#_Toc69214942)

[2.21. SSL Certificate Support 21](#_Toc69214943)

[2.22. LDAP Support 22](#_Toc69214944)

[2.23. Redfish Support 23](#_Toc69214945)

[3. BMC Internal Tool Usage 25](#_Toc69214946)

[3.1. GPIO Utilities 25](#_Toc69214947)

[3.2. I2C Tools 27](#_Toc69214948)

[3.3. obmc-console 28](#_Toc69214949)

[3.4. Accessing Devices from U-boot 29](#_Toc69214950)

[4. Supported IPMI Commands 30](#_Toc69214951)

# Introduction

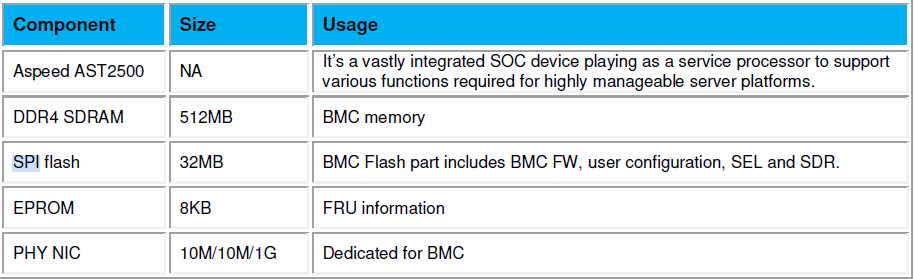
This document describes the supported feature in AMI OpenBMC and how to use the features.

TiogaPass hardware block diagram:

|  |
| --- |
|  |

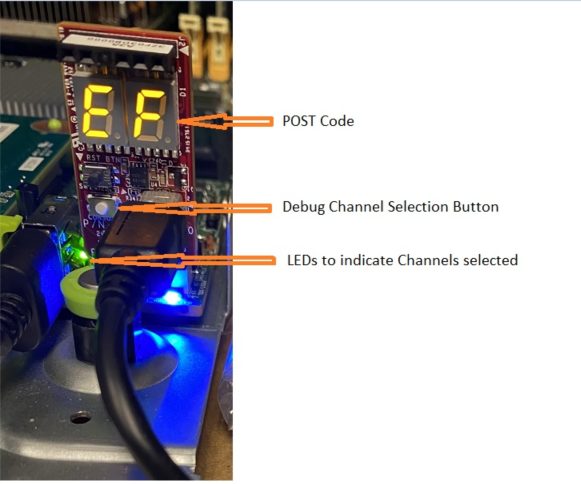
# Supported Features in OpenBMC

## BMC Hardware Component

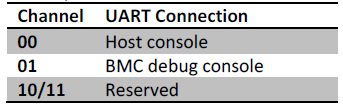


## BMC UART Console Access

TiogaPass uses a 14pin V1 debug card header for both BMC and BIOS/Host-OS serial output channel. There is a LCD on the header to display the BIOS POST Code. By default, TiogaPass selects BIOS/Host-OS output channel on the header. User should use the UART Channel selection button on the debug header to change the channel to be connected to BMC debug console.



Please refer to the following table for UART channel and connection



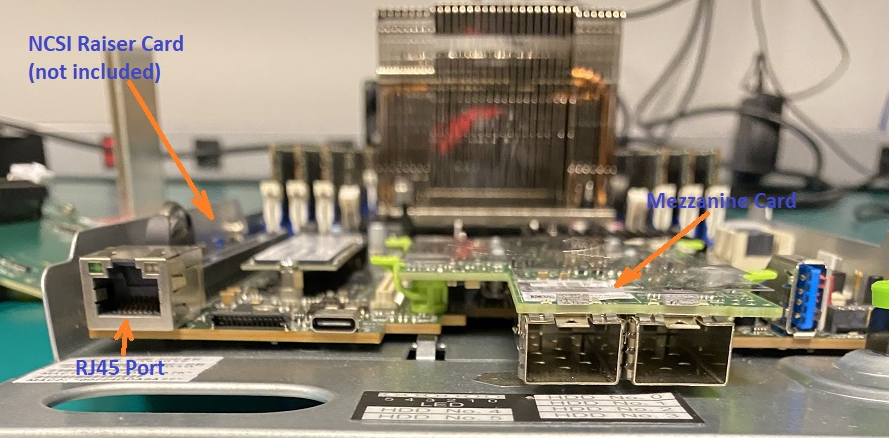
The settings of the UART ports, for both BMC and BIOS, are:

baud rate 115200, no flow control, terminal type VT100, 8 data bits, No Parity, 1 Stop Bit.

## BMC SPI Flash

The three onboard SPI Flash chips are used for holding Primary and Secondary BMC images, and BIOS image. Winbond W25Q256/257 and Numonyx N25Q256 are default supported SPI flashes, other brands are also supported but not fully tested.

## Network Interfaces



The TiogaPass platform uses Option 3 of the Management Network Interface from OCP spec. Intel I210-AT NIC used for RMII/NCSI interfaces, which passes both BMC and management data traffic with a RJ45 port. There is a Mezzanine Card which compatible with Intel v2.0 Motherboard, which holds two SFP interfaces with up to 10G data rate. There is also PCIe slot can be used for NCSI Riser Card, to hold the second NCSI interfaces.

## LAN Interface

Supports dedicated and shared LAN interface for Out-of-Band access to the Service Processor.

By default MAC will be auto generated and configured if it is not configured by the user. Supported IPMI commands are listed in section 4.

**Example:**

Ipmitool -H <BMC IP> -U root -I lanplus -P 0penBmc raw 0x6 0x1 🡪 Get Device ID

ipmitool -H <BMC IP> -U root -I lanplus -P 0penBmc lan print 1 🡪 Get Lan Info

ipmitool -H <BMC IP> -U root -I lanplus -P 0penBmc chassis power <on/off/cycle/reset/status>

ipmitool -H <BMC IP> -U root -I lanplus -P 0penBmc sensor

ipmitool -H <BMC IP> -U root -I lanplus -P 0penBmc sdr

## KCS Interface

KCS interface is supported in the firmware for In-band interface for Host to BMC communication.

Firmware supports only SMS interface, SMM interface not supported.

**Example:** ipmitool raw 6 1

## Sideband Interface

BMC supports the sideband network interface function. Shared lan between BMC and Host. By default the ETH0 is the sideband interface while the ETH1 is the dedicated network interface.It varies based on platform design. The following command can be tested using the either of the IP addresses.

**Example:** ipmitool -H <BMC IP> -I lanplus -U root -P 0penBmc raw 0x6 0x1

## Sensor Support

Sensor configuration and monitoring will be done through entity manager. Temperature, Voltage and Fan sensors are supported. It can be accessible through REST Over D-bus and IPMI.

Sensor Reading Through IPMI:

# ipmitool -H <IP> -U root -P 0penBmc -I lanplus sensor

Chassis Intrusio | 0x0 | discrete | 0x0000| na | na | na | na | na | na

ACPI State | 0x0 | discrete | 0x0000| na | na | na | na | na | na

SEL | 0x0 | discrete | 0x0000| na | na | na | na | na | na

WATCHDOG2 | 0x0 | discrete | 0x0000| na | na | na | na | na | na

MB FAN0 TACH | 9212.000 | RPM | nc | na | 490.000 | na | 8526.000 | 11466.000 | na

MB FAN1 TACH | 3332.000 | RPM | ok | na | 490.000 | na | 8526.000 | 11466.000 | na

MB INLET REMOTE | 39.000 | degrees C | ok | na | na | na | na | na | na

MB INLET TEMP | 45.000 | degrees C | cr | na | na | na | na | 40.000 | na

MB OUTLET REMOTE | 32.000 | degrees C | ok | na | na | na | na | na | na

MB OUTLET TEMP | 33.000 | degrees C | ok | na | na | na | na | 90.000 | na

MEZZ SENSOR REMO | 0.000 | degrees C | ok | na | na | na | na | 95.000 | na

MEZZ SENSOR TEMP | 91.000 | degrees C | ok | na | na | na | na | na | na

MB P3V BAT | 3.138 | Volts | ok | na | 2.735 | na | na | 3.731 | na

MB P1V05 | 1.044 | Volts | ok | na | 0.937 | na | na | 1.150 | na

MB P3V3 | 3.302 | Volts | ok | na | 2.976 | na | na | 3.612 | na

MB P3V3 STBY | 3.302 | Volts | ok | na | 2.976 | na | na | 3.612 | na

MB P5V | 4.940 | Volts | ok | na | 4.524 | na | na | 5.486 | na

MB P5V STBY | 4.940 | Volts | ok | na | 4.524 | na | na | 5.486 | na

MB P12V | 11.970 | Volts | ok | na | 10.773 | na | na | 13.230 | na

MB PVNN PCH STB | 0.994 | Volts | ok | na | 0.760 | na | na | 1.101 | na

## System Event Log Support

BMC FW supports system event logging. SEL list and add can be done using the standard IPMI commands.

**For example:** ipmitool -U root -P 0penBmc -I lanplus -H <BMC IP> sel list

In the **Server Health** option, click **System log** tab where SEL can be viewed on web UI as shown below.

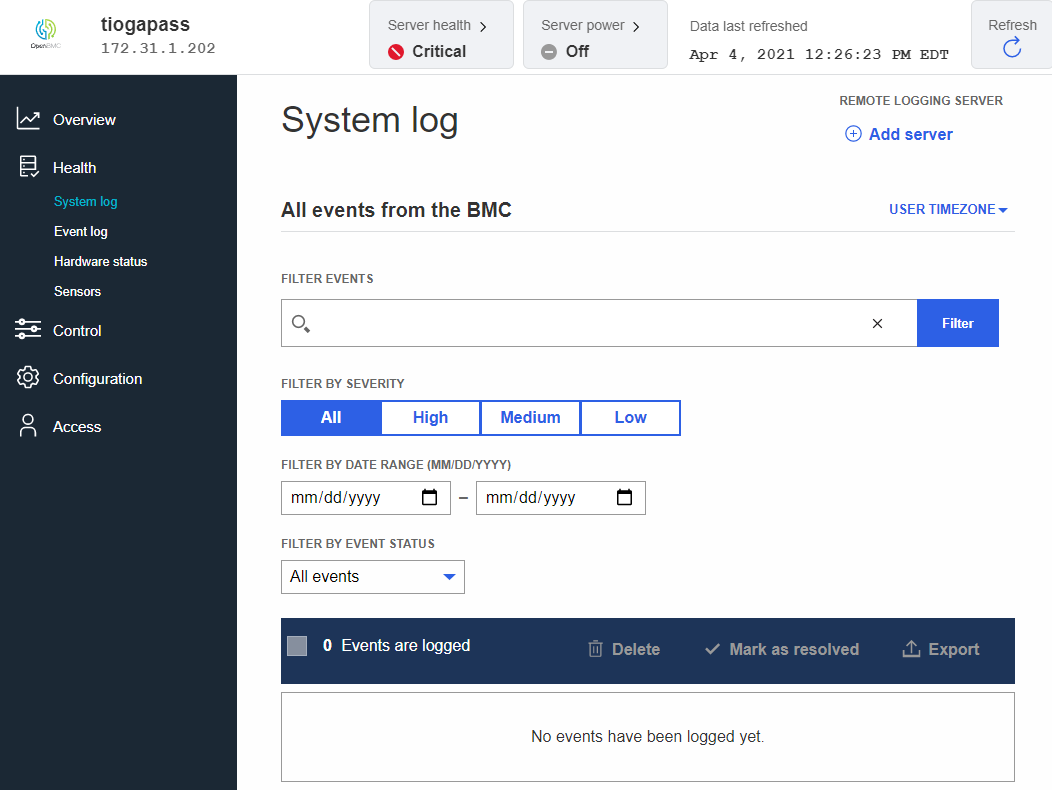


Figure 1 System Log

## FRU Device Support

BMC FW supports the IPMI FRU devices support. FRU device configuration can be done through Entity manager, read and write FRU data to eeprom are supported.

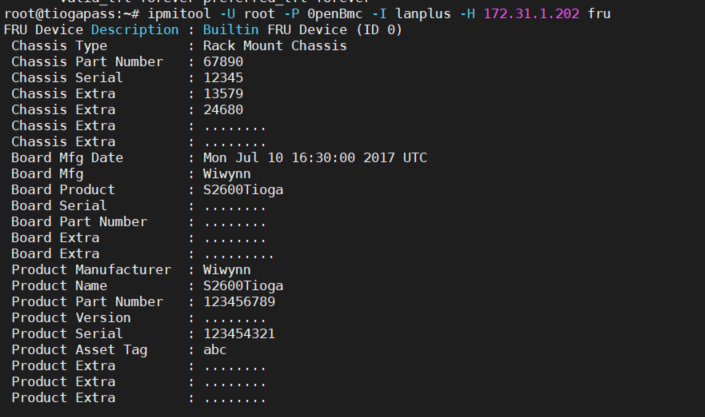


Figure 2 FRU Device Support

## BMC Firmware Upgrade Support

Four types of flashing methods are supported:

1. Soc Flash
2. Dediprog
3. TFTP flash
4. Web UI flash

Please refer Getting Started Guide for more details on each of the methods.

## WebUI Support

BMC FW supports a web server based on the bmcweb implementation and phosphor web UI front end. This can be accessed using the BMC IP address. A sample screenshot of the login page is given below.

1. Use valid user credentials (default are root/0penBmc) to login.

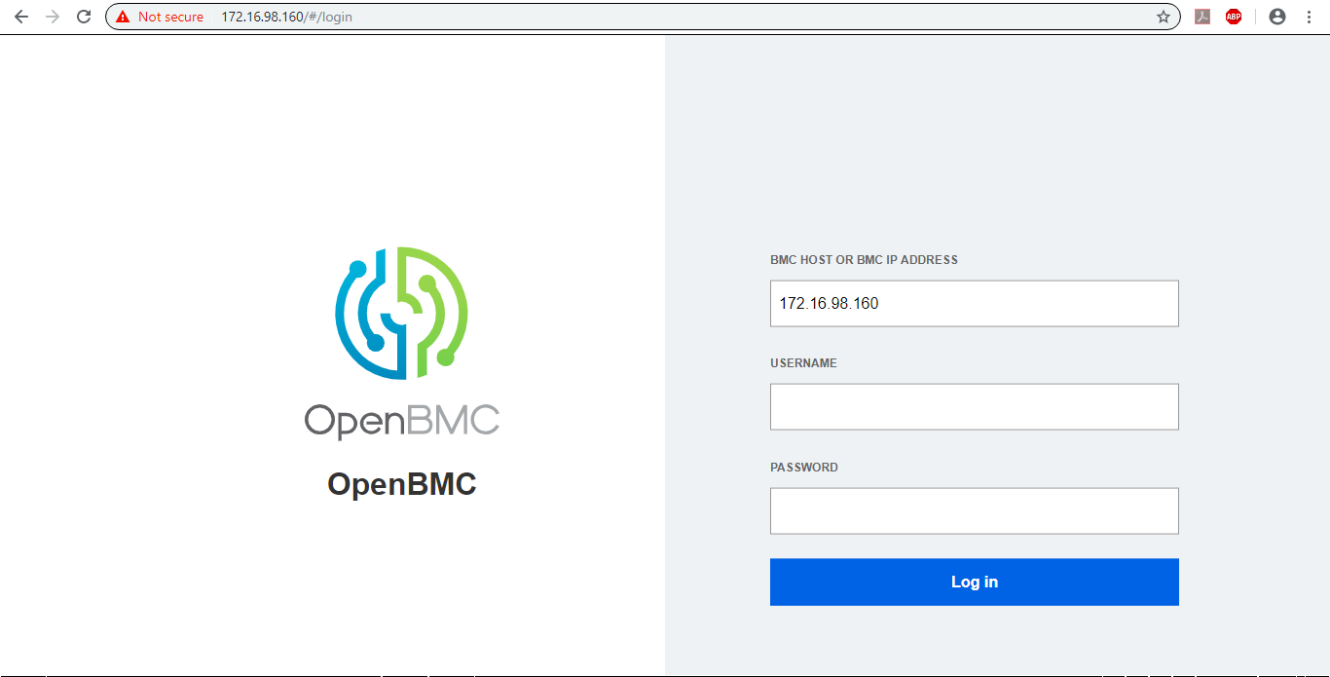


Figure 3 Web UI support

1. Once logged in the default idle screen as shown below.

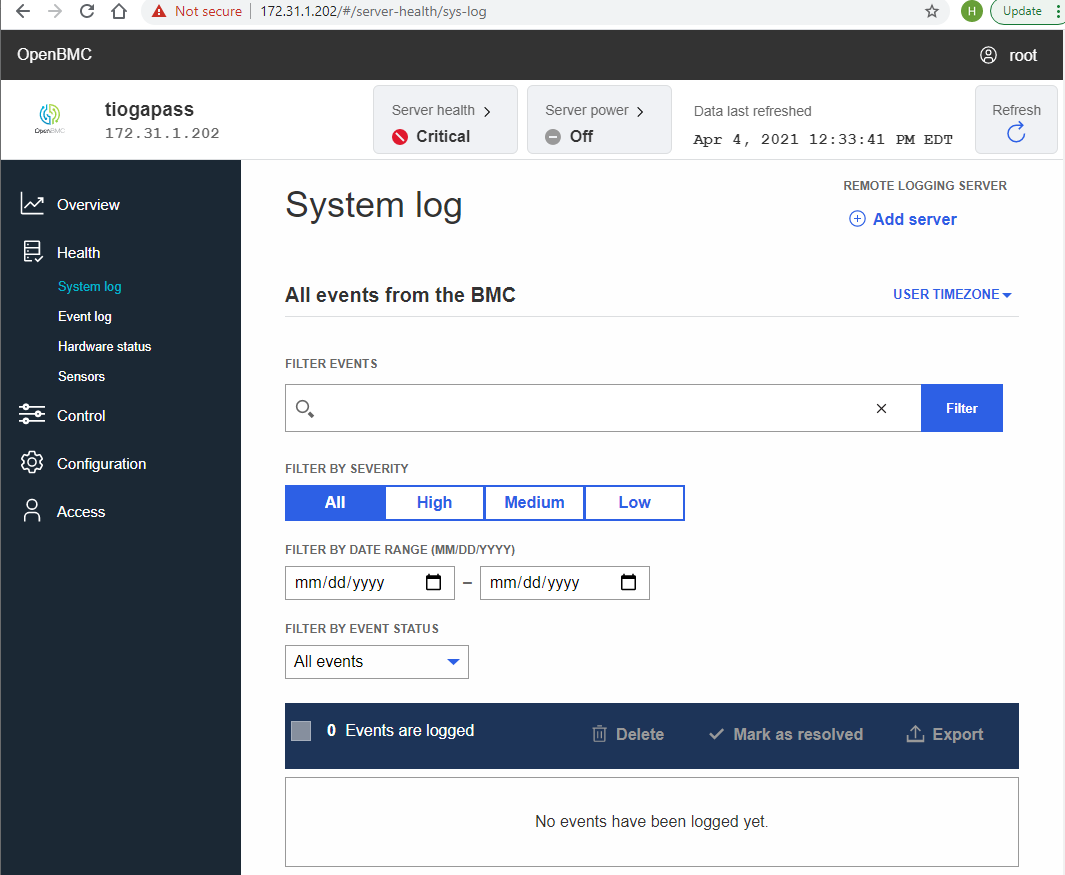


Figure 4 Default Idle Screen

1. Each of the options on the left panel opens up multiple other related options for the user. For example **Server Health** option displays the Event log, Hardware status, Sensors and System logs. For instance, Sensor page is displayed as below. This has all the sensors showing with their current readings and status.

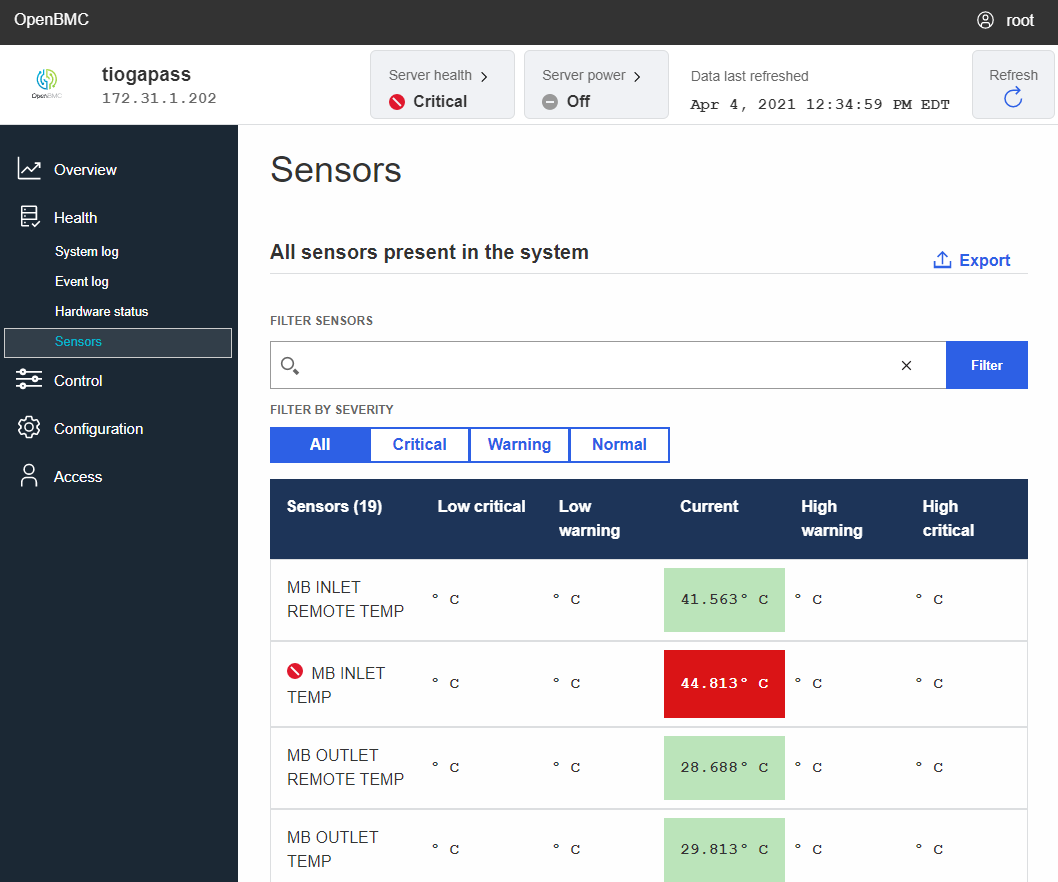


Figure 5 Sensor

## KVM Support

BMC FW provides support to access the server console via KVM. Click **Server Control** 🡪 **KVM**, to access the KVM. A sample screenshot is displayed below.

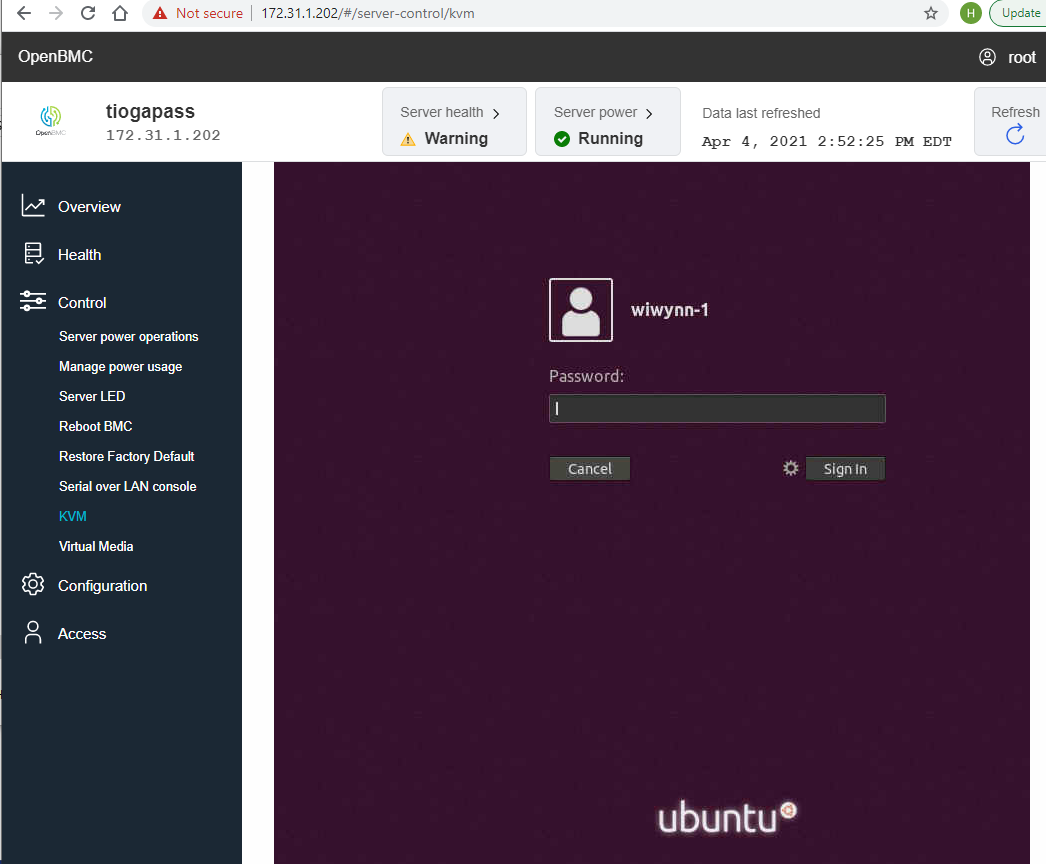


Figure 6 KVM Support

## Reboot BMC

We can reboot the BMC through web UI. Click **Server Control** 🡪 **Reboot BMC**, to reboot the BMC.

A sample screenshot is displayed below.

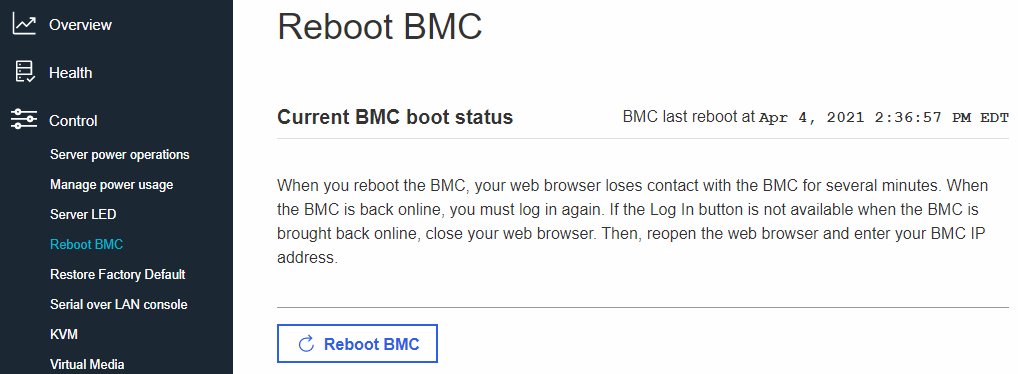


Figure 7 Reboot BMC

## Server Power Operation

Chassis power operation are supported through web UI. Click **Server Control** 🡪 **Server Power Operation**, to reboot the BMC. A sample screenshot is displayed below.

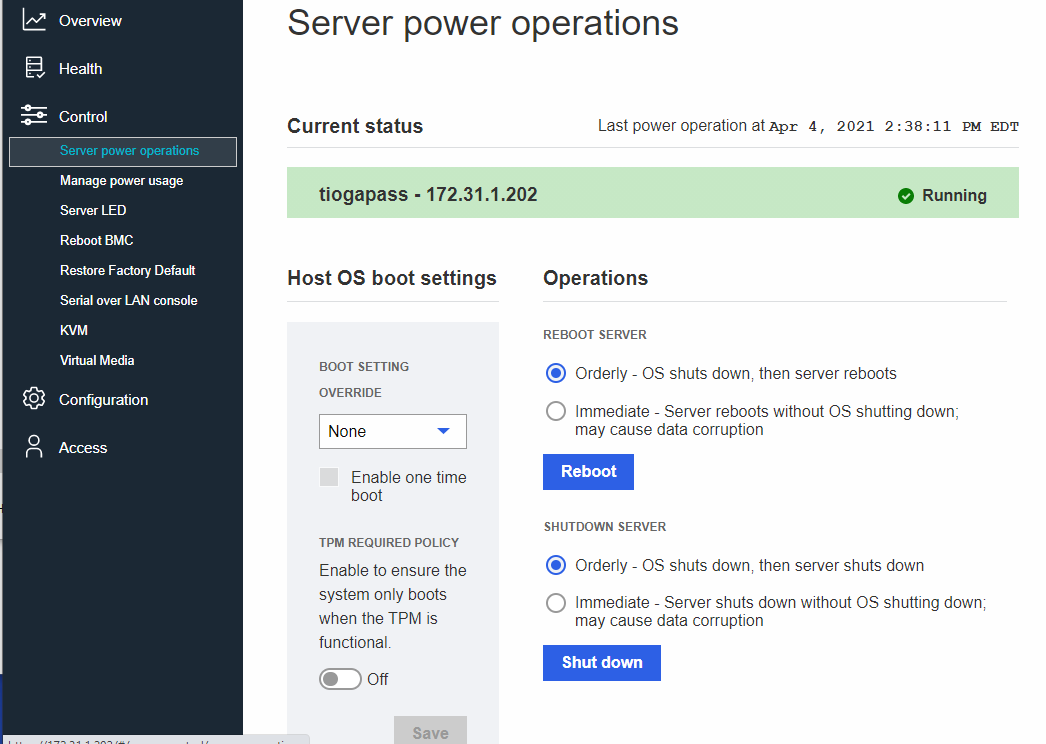


Figure 8 Reboot BMC

## NTP Support

NTP support for time setting is available using the WebUI. After login with the valid user credentials (root/0penBmc), click **Server configuration** > **Date and Time setting** option. BMC time can be set either manually or automatically using NTP and the specified NTP server. One or more servers can be added using the button as shown below.

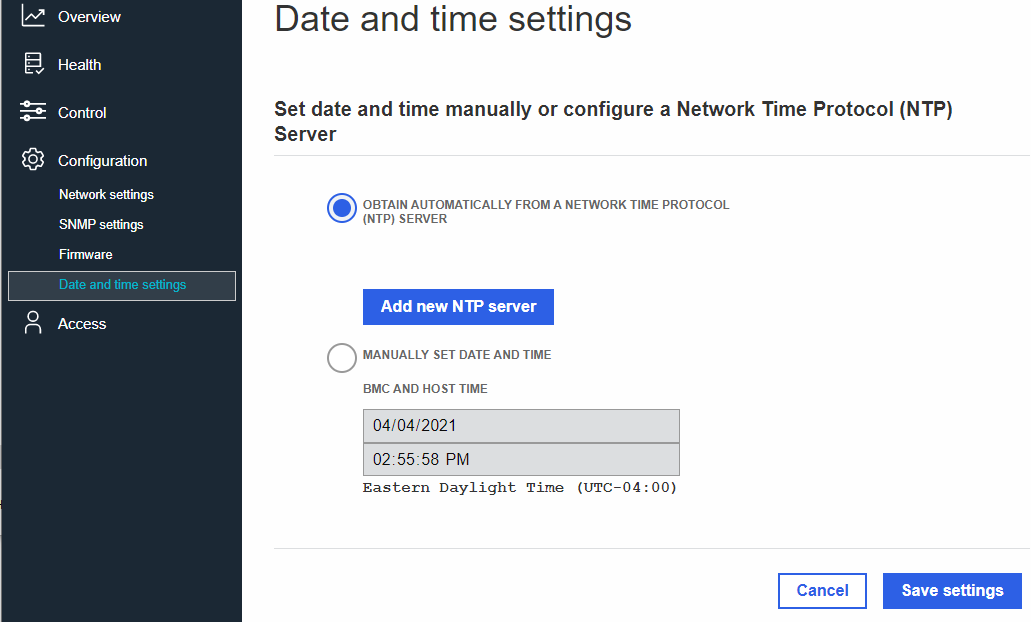


Figure 9 NTP Support

## BMC Network Settings

BMC Web UI supports option to modify and edit the BMC network settings. On the default login page, select the BMC Network Setting option to access this page as shown below. The same can be accessed by clicking **Server configuration** 🡪 **Network Settings**.

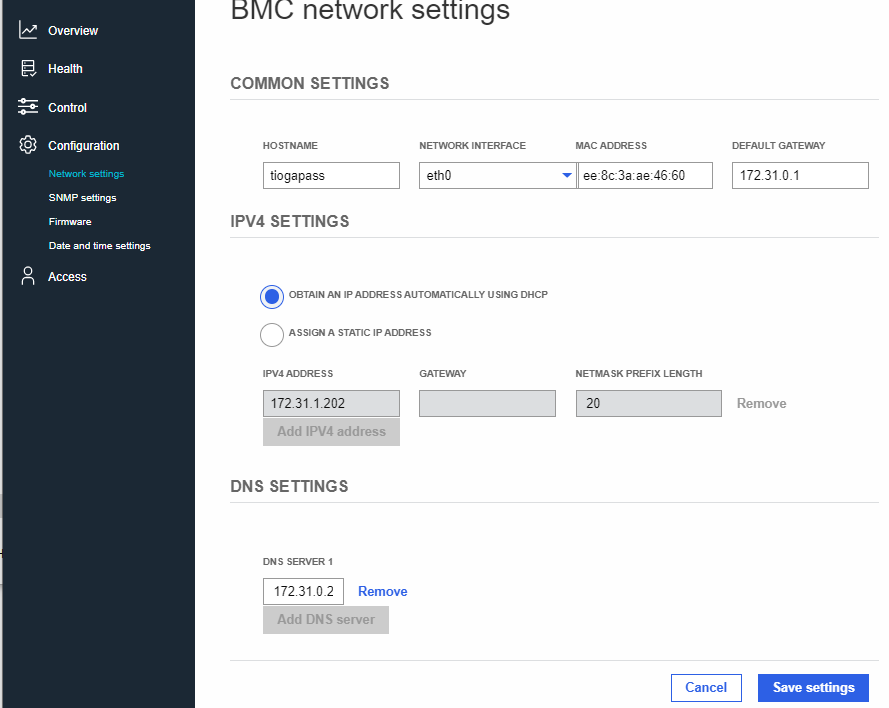


Figure 10 BMC Network settings

## User Management Support

BMC FW supports User management including adding a new user, deleting an existing user, modifying the privileges of an existing user, etc. This can be done using the Access Control option on the left side panel of the BMC web UI. Select Local users on the extended left panel to view the page and manage users.

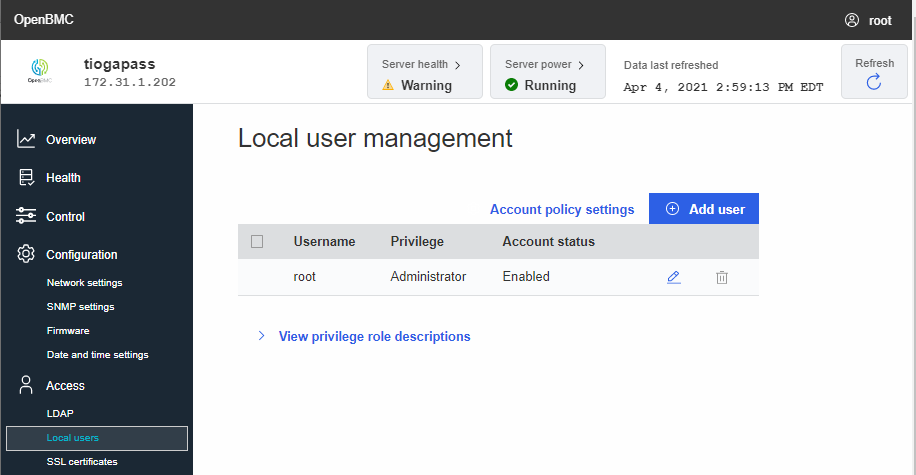


Figure 11 User Management support

LDAP support is currently disabled on this release of BMC FW.

## IPMI 2.0 and DCMI 1.5 support:

IPMI 2.0 based implementation is supported on the BMC FW. Not all the mandatory commands are implemented in the current version of the FW. The list of supported commands is listed in section 4. IPMI based host power controls are supported. Please refer release document for known issues.

DCMI 1.5 capability is supported on this release of the BMC FW. The list of supported commands is specified in section 4. Please refer release documentation for details on each of supported commands.

## vMedia Support

BMC FW supports virtual media. User navigates to “Server control 🡪 Virtual Media” page, after logging in on the webUI. As shown below there will be option to choose a file.

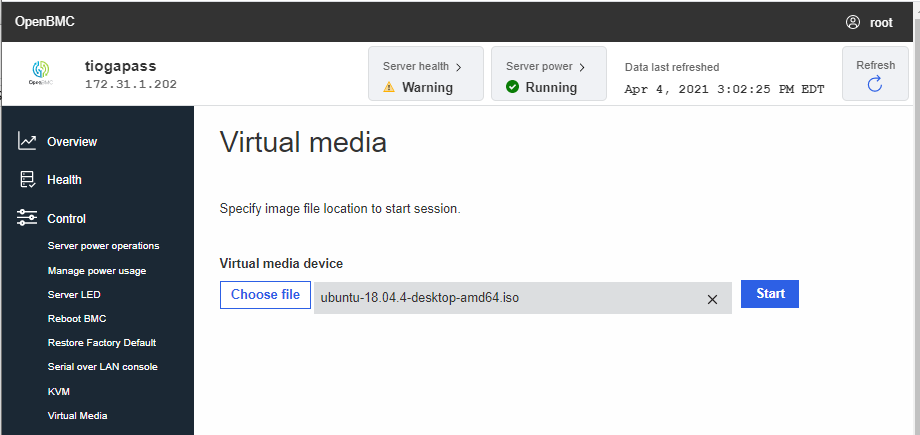


Figure 12 vMedia Support

User selects file and pushes 'start' button to establish websocket connection. Then you see the following page showing active session.

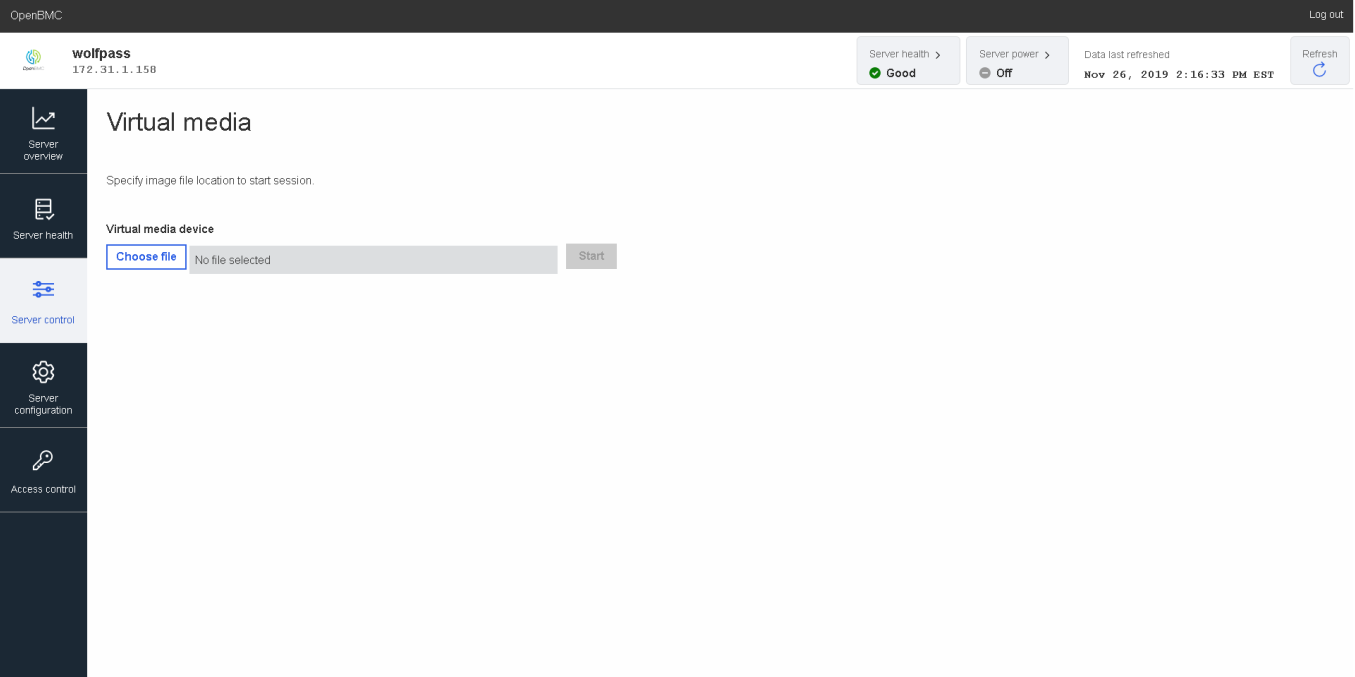
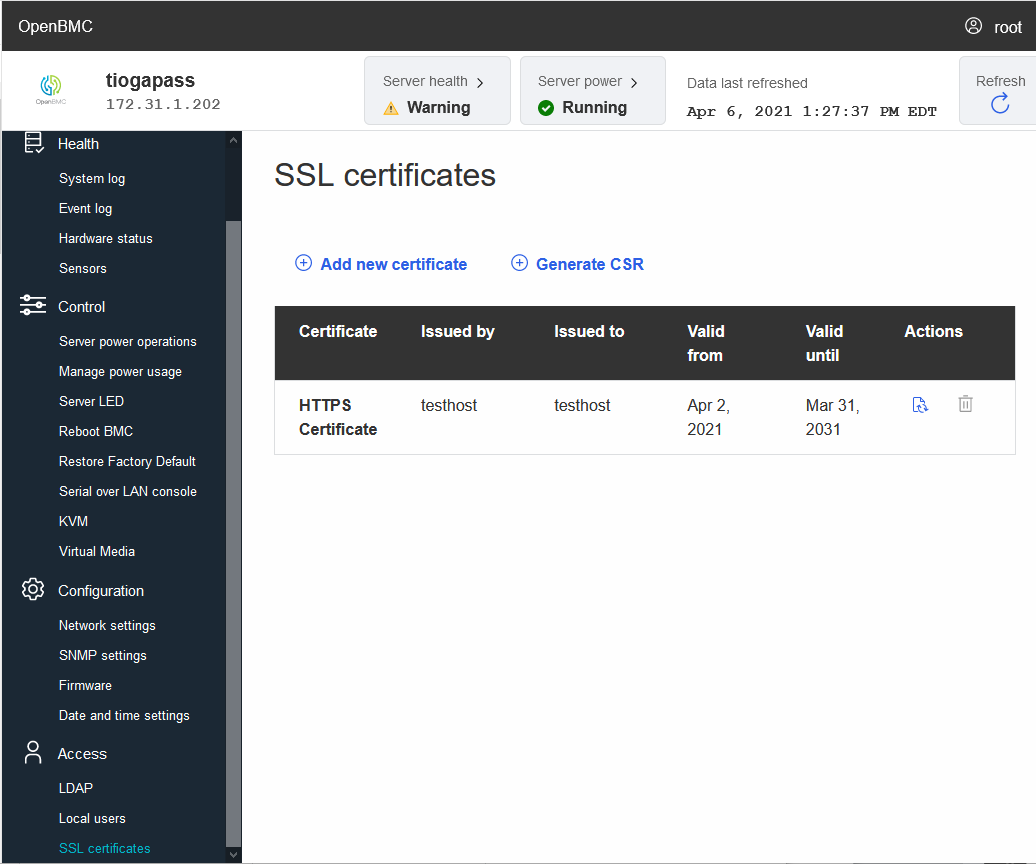


Figure 13 Virtual Media

The nbdServerService on FW provide ability for user to navigate away from the page and return with the ability to see the current active sessions. For example, you can upload Ubuntu image file and mount virtual media device from host console. Then you will be able to see Ubuntu image file. Finally, when connection is stopped, 'USB disconnect' log would be appear on host console.

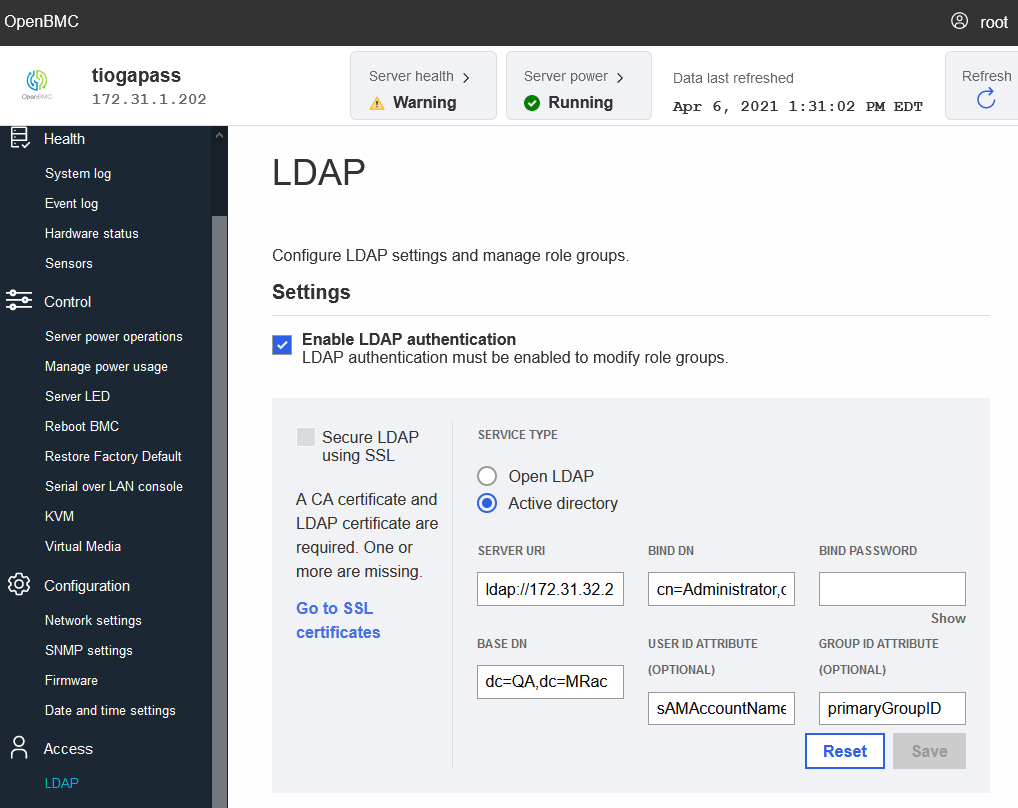
## SSL Certificate Support

BMC FW supports SSL Certificate. User navigates to “Access 🡪 SSL Certificate” page, after logging in on the webUI. As shown below there will be options to add new certificate, generate CSR.



## LDAP Support

BMC FW supports configuring LDAP settings and manage role groups.



## Redfish Support

BMC FW supports Redfish, here are some examples:

***GET Schema Init -******http://{{ip}}/redfish/v1/***

|  |
| --- |
|  |

**System Collection - http://{{ip}}/redfish/v1/Systems/**

|  |
| --- |
|  |

# BMC Internal Tool Usage

## GPIO Utilities

There are a few GPIO utilities supported on the BMC FW. They can be used to set/get state on the GPIO lines, set/get direction of a GPIO line etc.

1. **gpiodetect:**

Usage: gpiodetect

List all GPIO chips; print their labels and number of GPIO lines.

**Example:**

root@wolfpass:~# gpiodetect

gpiochip0 [1e780000.gpio] (232 lines)

1. **gpiofind:**

Usage: gpiofind <name>

Find a GPIO line by name. The output of this command can be used as input for gpioget/set.

**Example:**

root@wolfpass:~# gpiofind SIO\_S3

gpiochip0 192

1. **gpioset:**

Usage: gpioset [OPTIONS] <chip name/number> <offset1>=<value1> <offset2>=<value2> ...

Set GPIO line values of a GPIO chip and maintain the state until the process exits.

**Options:**

-v, --version: display the version and exit

-l, --active-low: set the line active state to low

-m, --mode=[exit|wait|time|signal] (defaults to 'exit'):

tell the program what to do after setting values

-s, --sec=SEC: specify the number of seconds to wait (only valid for --mode=time)

-u, --usec=USEC: specify the number of microseconds to wait (only valid for --mode=time)

-b, --background: after setting values: detach from the controlling terminal

**Modes:**

exit: set values and exit immediately

wait: set values and wait for user to press ENTER

time: set values and sleep for a specified amount of time

signal: set values and wait for SIGINT or SIGTERM

Note: The state of a GPIO line controlled over the character device reverts to default when the last process referencing the file descriptor representing the device file exits. This means that it's wrong to run gpioset, have it exit and expect the line to continue being driven high or low. It may happen if given pin is floating but it must be interpreted as undefined behavior.

1. **gpioget:**

**Usage:** gpioget [OPTIONS] <chip name/number> <offset 1> <offset 2> ...

Read line value(s) from a GPIO chip

**Options:**

-v, --version: display the version and exit

-l, --active-low: set the line active state to low

**Example:**

root@wolfpass:~# gpioget gpiochip0 2

0

1. **gpioinfo:**

**Usage:** gpioinfo [OPTIONS] <gpiochip1>..

Print information about all lines of the specified GPIO chip(s) (or all gpiochips if none are specified).

1. **gpiomon:**

**Usage:** gpiomon [OPTIONS] <chip name/number> <offset 1> <offset 2> ...

Wait for events on GPIO lines and print them to standard output

**Options:**

-v, --version: display the version and exit

-l, --active-low: set the line active state to low

-n, --num-events=NUM: exit after processing NUM events

-s, --silent: don't print event info

-r, --rising-edge: only process rising edge events

-f, --falling-edge: only process falling edge events

-b, --line-buffered: set standard output as line buffered

-F, --format=FMT specify custom output format

**Format Specifiers:**

%o: GPIO line offset

%e: event type (0 - falling edge, 1 rising edge)

%s: seconds part of the event timestamp

%n: nanoseconds part of the event timestamp

## I2C Tools

1. **i2cdetect:**

Usage: i2cdetect [-y] [-a] [-q|-r] I2CBUS [FIRST LAST]

i2cdetect -F I2CBUS

i2cdetect -l

I2CBUS is an integer or an I2C bus name

If provided, FIRST and LAST limit the probing range.

1. **i2cdump:**

Usage: i2cdump [-f] [-y] [-r first-last] [-a] I2CBUS ADDRESS [MODE [BANK [BANKREG]]]

I2CBUS is an integer or an I2C bus name

ADDRESS is an integer (0x03 - 0x77, or 0x00 - 0x7f if -a is given)

MODE is one of:

b (byte, default)

w (word)

W (word on even register addresses)

s (SMBus block)

i (I2C block)

c (consecutive byte)

d (double word)

Append p for SMBus PEC

1. **i2cget:**

Usage: i2cget [-f] [-y] [-a] I2CBUS CHIP-ADDRESS [DATA-ADDRESS [MODE] [LENGTH]]

I2CBUS is an integer or an I2C bus name

ADDRESS is an integer (0x03 - 0x77, or 0x00 - 0x7f if -a is given)

MODE is one of:

b (read byte data, default)

w (read word data)

c (write byte/read byte)

i (read I2C block data)

Append p for SMBus PEC

LENGTH is length for block data reads

1. **i2cset:**

Usage: i2cset [-f] [-y] [-m MASK] [-r] [-a] I2CBUS CHIP-ADDRESS DATA-ADDRESS [VALUE] ... [MODE]

I2CBUS is an integer or an I2C bus name

ADDRESS is an integer (0x03 - 0x77, or 0x00 - 0x7f if -a is given)

MODE is one of:

c (byte, no value)

b (byte data, default)

w (word data)

i (I2C block data)

s (SMBus block data)

Append p for SMBus PEC

## obmc-console

obmc-console is used to redirect serial (UART) over ssh. We can specify host serial port in obmc-console-server/systemd service. There are two methods to get remote host console:

* Standalone tool from bmc: obmc-console-client
* Using ssh: ssh -p2200 root@<BMC IP>

## Accessing Devices from U-boot

1. **Read memory:**

md <memory register offset address>

Example: Display USB1.1 HID Controller Function Control and Status

md 0x1e6e1000

1. **Write memory:**

mw <memory register offset address> <value>

Example: Display USB1.1 HID Controller Function Control and Status:

md 0x1e6e1000 0x00000003 🡪 To enable USB connection and Low speed.

**Note:** The above command will apply the 32 bit value into memory offset.

Please refer AST data sheet for address mapping and register offset.

# Supported IPMI Commands

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Net Function** |  | **Command** | **Command Name** | **Comments** |
| Application |  |  |  |  |
| 0x06 |  | 0x01 | Get Device ID |  |
| 0x06 |  | 0x02 | Cold Reset |  |
| 0x06 |  | 0x08 | Get Device GUID |  |
| 0x06 |  | 0x38 | Get Channel Authentication Capabilities |  |
| 0x00 |  | 0x01 | Get Chassis Status |  |
| 0x00 |  | 0x09 | Get System Boot Options | Only parameter 5 is supported(as per IPMI Spec ) |
| 0x00 |  | 0x0F | Get POH Counter |  |
| SensorEvent |  |  |  |  |
| 0x04 |  | 0x20 | Get Device SDR Info |  |
| 0x04 |  | 0x21 | Get Device SDR |  |
| 0x04 |  | 0x22 | Reserve Device SDR Repository |  |
| 0x04 |  | 0x2D | Get Sensor Reading |  |
| Storage |  |  |  |  |
| 0x0a |  | 0x10 | Get FRU Inventory Area Info |  |
| 0x0a |  | 0x11 | Read FRU Data |  |
| 0x0a |  | 0x12 | Write FRU Data |  |
| 0x0a |  | 0x20 | Get SDR Repository Info |  |
| 0x0a |  | 0x23 | Get SDR |  |
| 0x0a |  | 0x40 | Get SEL Info |  |
| 0x0a |  | 0x44 | Add SEL Entry |  |
| 0x0a |  | 0x43 | Get SEL Entry |  |
| 0x0a |  | 0x42 | Reserve SEL |  |
| 0x0a |  | 0x47 | Clear SEL |  |
| 0x0a |  | 0x48 | Get SEL Time |  |
| 0x0a |  | 0x49 | Set SEL Time |  |
| Transport |  |  |  |  |
| 0x0c |  | 0x01 | Set LAN Configuration Parameters | SUPPORTED [Authentication type, ip-address, ip-address-source, MAC address, subnet-mask] |
| 0x0c |  | 0x02 | Get LAN Configuration Parameters | SUPPORTED [Authentication type ,ip-address, ip-address-source, MAC address, subnet-mask] |
| 0x0c |  | 0x21 | Set SOL Configuration Parameters | SOL Payload Port Number, SOL Payload Channel, SOL retry, SOL Authentication not supported |
| 0x0c |  | 0x22 | Get SOL Configuration Parameters | SOL retry, SOL Authentication not supported |
| DCMI |  |  |  |  |
| 0x2c |  | 0x01 | Get DCMI Capability Info | parameter 5 is not supported, will get default values back |
| 0x2c |  | 0x03 | Get Power Limit |  |
| 0x2c |  | 0x04 | Set Power Limit | Exception Actions, Correction Time Limit, Sampling period not working |
| 0x2c |  | 0x05 | Activate Power Limit |  |
| 0x2c |  | 0x06 | Get Asset Tag |  |
| 0x2c |  | 0x08 | Set Asset Tag |  |
| 0x2c |  | 0x09 | Get Management Controller Id String |  |
| 0x2c |  | 0x10 | Get Temperature Reading |  |
| 0x2c |  | 0x12 | Set DCMI Configuration Parameters | SUPPORTED [Activate DHCP] |
| 0x2c |  | 0x13 | Get DCMI Configuration Parameters | SUPPORTED [Activate DHCP] |

|  |  |
| --- | --- |
| lan | print, set, alert print, alert set, stats clear |
| chassis | status, power, policy, poh, bootdev, bootparam |
| power | status, on, off, reset |
| event |  |
| mc | reset, cold, info |
| sdr |  |
| sensor |  |
| fru | print, read, write |