

Brad Chou / openbmc

Source

5aec584647f

...

openbmc / meta-hdsc / meta-hs2500 / doc / RunBMC\_HS2500\_Quick\_Start\_Guide.md

Brad Chou committed 5aec584647f 1 week ago

Source view

Diff to previous

History

Contributors

Show source

Edit

Blame

Raw file

# RunBMC HS2500 Quick Start Guide

RunBMC HS2500 is an OCP project that use OpenBMC as it's internal BMC management firmware. This quick start guide will teach you how to do basic function test within BMC console.

## BCM Console Login

Connect the micro usb uart5 to your host PC, it will show up a new serial device to you PC. Use any terminal emulator program such as 'putty' or 'minicom' to connect this serial device with baud rate 115200. You will see the login prompt for openbmc console :

```
Phosphor OpenBMC (Phosphor OpenBMC Project Reference Distro) 0.1.0 hs2500 ttyS4

hs2500 login:
```

By default the login is 'root', password is 'OpenBmc'

```
Phosphor OpenBMC (Phosphor OpenBMC Project Reference Distro) 0.1.0 hs2500 ttyS4

hs2500 login: root
Password: OpenBmc
root@hs2500:~#
```

## Set MAC address

HS2500 has no default MAC address. You can set it manually. When there is no MAC address saved, openbmc will generate a random address every time.

```
fw_setenv ethaddr xx:xx:xx:xx:xx:xx
fw_setenv ethladdr xx:xx:xx:xx:xx:xx
reboot
```

After BMC reboot, the MAC address will apply.

## BMC network IP address

By default, BMC use DHCP to get IP address from DHCP server. The 'ip address' command will display all ip address.

```
root@hs2500:~# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:11:22:33:44:77 brd ff:ff:ff:ff:ff:ff
    inet 169.254.233.135/16 brd 169.254.255.255 scope link eth0
        valid_lft forever preferred_lft forever
    inet 192.168.1.74/24 brd 192.168.1.255 scope global dynamic eth0
        valid_lft 86374sec preferred_lft 86374sec
    inet6 fe80::211:22ff:fe33:4477/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
```

```
link/ether 00:11:22:33:44:66 brd ff:ff:ff:ff:ff:ff
inet 169.254.229.85/16 brd 169.254.255.255 scope link eth1
    valid_lft forever preferred_lft forever
inet 192.168.1.57/24 brd 192.168.1.255 scope global dynamic eth1
    valid_lft 86373sec preferred_lft 86373sec
inet6 fe80::211:22ff:fe33:4466/64 scope link
    valid_lft forever preferred_lft forever
4: sit0@NONE: <NOARP,UP,LOWER_UP> mtu 1480 qdisc noqueue qlen 1000
    link/sit 0.0.0.0 brd 0.0.0.0
```

OpenBMC support IP aliases, so you can assign multiple IP address to a network interface.  
For example, to add another IP "192.168.1.75" to eth0.

```
root@hs2500:~# ip address add 192.168.1.75/24 dev eth0
root@hs2500:~# ip address show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:11:22:33:44:55 brd ff:ff:ff:ff:ff:ff
    inet 169.254.87.164/16 brd 169.254.255.255 scope link eth0
        valid_lft forever preferred_lft forever
    inet 192.168.1.74/24 brd 192.168.1.255 scope global dynamic eth0
        valid_lft 85432sec preferred_lft 85432sec
    inet 192.168.1.75/24 scope global secondary eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::211:22ff:fe33:4455/64 scope link
        valid_lft forever preferred_lft forever
```

## BMC firmware update

BMC firmware can be updated if there is a file named 'image-bmc' in the /run/initramfs/ directory.  
You can copy BMC firmware by scp command from your host.  
The file name must be called **image-bmc**

```
export server=10.19.84.81
cd /run/initramfs/
scp user@{server}:image-bmc ./
reboot
```

During the BMC reboot process, the firmware will be updated.

## LED lamp\_test

lamp\_test will blink all 7 segments LEDs. It is useful to verify if all LEDs are workable.

```
busctl call `mapper get-service /xyz/openbmc_project/led/groups/lamp_test` \
/xyz/openbmc_project/led/groups/lamp_test org.freedesktop.DBus.Properties \
Set ssv xyz.openbmc_project.Led.Group Asserted b true
```

## Buttons

Before doing power on/off test, connect J2002 pin #3 and pin #4 together by a jumper to let BMC sense correct power status.  
There are 8 buttons on the HS2500, as described here:

Button	Name	Action
SW701	ID	Toggle to identify LED
SW702	Reset	Toggle to reset host
SW703	Power	Toggle host power on / off
		Long press to force power off
SW704	LED Test	Blink all LEDs
SW705	Fan Test	Toggle FAN PWM between 50% and 100% duty

## I2C Device Detect

There are 12 I2C buses on the HS2500. The 'i2cdetect' can scan all available devices on a given i2c bus and display the slave address. For example, to detect devices on i2c10

```
root@hs2500:~# i2cdetect -y 10
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  -- 21 --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  -- 38 --  --  --  --  --  --  --  --
40: 40 41 --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

All available slave devices are described in this table.

Bus Number	Slave Address	Device Name	EVT Build
0	N/A	N/A	
1	N/A	N/A	
2	N/A	N/A	
3	50	AT24C64D-SSHM-T	
4	N/A	N/A	
5	N/A	N/A	
6	N/A	N/A	
7	N/A	N/A	
8	N/A	N/A	
9	20	TPM SLB9645TT12FW13332XUMA1	
10	21	TCA9555	
	38	PCA9554	
	40	INA219AID	
	41	INA219AID	
11	48	TMP75AIDGK	
	70	PCA9548	
12	4a	TMP75AIDGK	
	51	M24C64-RDW6TP	

Bus Number	Slave Address	Device Name	PVT Build
0	N/A	N/A	
1	N/A	N/A	
2	N/A	N/A	
3	50	AT24C64D-SSHM-T	
4	N/A	N/A	
5	N/A	N/A	
6	N/A	N/A	
7	N/A	N/A	
8	4a	TMP75AIDGK	

Bus Number	Slave Address	Device Name	PVT Build
	51	M24C64-RDW6TP	
9	20	TPM SLB9645TT12FW13332XUMA1	
10	21	TCA9555	
	38	PCA9554	
	40	INA219AID	
	41	INA219AID	
11	48	TMP75AIDGK	
	70	PCA9548	
12	N/A	N/A	

## Read ADC Voltage

There are 8 ADC channels on HS2500.  
Channel 0 ~ 3 are connected to fixed voltage.  
Channel 4 ~ 7 are adjustable by divider.  
You can read ADC voltage from any channel. For example, to read voltage from ADC channel 2

```
root@hs2500:~# cat /sys/class/hwmon/hwmon1/in2_input
1694
```

ADC Channel	Voltage	Accuracy
in1_input	1798	+ - 18
in2_input	1694	+ - 18
in3_input	1126	+ - 18
in4_input	833	+ - 18
in5_input	0 ~ 1798	+ - 18
in6_input	0 ~ 1798	+ - 18
in7_input	0 ~ 1798	+ - 18
in8_input	0 ~ 1798	+ - 18

## Fan Tach and PWM

There 4 fan tach and PWM on the HSBUV. All of them can be read/write by sysfs in /sys/class/hwmon/hwmon0.  
For example, to read fan1 speed :

```
root@hs2500:~# cat /sys/class/hwmon/hwmon0/fan1_input
23993
```

To set the fan1 PWM to 50% duty :

```
root@hs2500:~# echo 127 > /sys/class/hwmon/hwmon0/pwm1
```

The available range of PWM is between 0 ~ 255, which is mapping to duty cycle 0% ~ 100%.

## OpenBMC REST API

The primary management interface for OpenBMC is REST. This document provides some basic structure and usage examples for the REST interface.  
The REST API is for BMC out of band management, so you should connect BMC to a network and send commands by curl from another host.

## Authentication

This tutorial uses the basic authentication URL encoding, so just pass in the user name and password as part of the URL and no separate login/logout commands are required:

```
export bmc=<username>:<password>@<bmcip>
```

## HTTP GET operations & URL structure

- To query the attributes of an object, perform a GET request on the objectname, with no trailing slash. For example:

```
$ curl -k https://{$bmc}/xyz/openbmc_project/user
{
  "data": {
    "AccountUnlockTimeout": 0,
    "AllGroups": [
      "ipmi",
      "redfish",
      "ssh",
      "web"
    ],
    "AllPrivileges": [
      "priv-admin",
      "priv-operator",
      "priv-user",
      "priv-callback"
    ],
    "MaxLoginAttemptBeforeLockout": 0,
    "MinPasswordLength": 8,
    "RememberOldPasswordTimes": 0
  },
  "message": "200 OK",
  "status": "ok"
}
```

- To query a single attribute, use the `attr/<name>` path. Using the `user` object from above, we can query just the `MinPasswodLength` value:

```
$ curl -k https://{$bmc}/xyz/openbmc_project/user/attr/MinPasswodLength
{
  "data": 8,
  "message": "200 OK",
  "status": "ok"
}
```

- When a path has a trailing-slash, the response will list the sub objects of the URL. For example, using the same object path as above, but adding a slash:

```
$ curl -k https://{$bmc}/xyz/openbmc_project/
{
  "data": [
    "/xyz/openbmc_project/Chassis",
    "/xyz/openbmc_project/Ipmi",
    "/xyz/openbmc_project/certs",
    "/xyz/openbmc_project/console",
    "/xyz/openbmc_project/control",
    "/xyz/openbmc_project/dump",
    "/xyz/openbmc_project/events",
    "/xyz/openbmc_project/inventory",
    "/xyz/openbmc_project/ipmi",
    "/xyz/openbmc_project/led",
    "/xyz/openbmc_project/logging",
    "/xyz/openbmc_project/network",
    "/xyz/openbmc_project/object_mapper",
    "/xyz/openbmc_project/software",
    "/xyz/openbmc_project/state",
    "/xyz/openbmc_project/time",
    "/xyz/openbmc_project/user"
  ],
  "message": "200 OK",
  "status": "ok"
}
```

- Performing the same query with `/list` will list the child objects *recursively*.

```
$ curl -k https://$bmc}/xyz/openbmc_project/network/list
{
  "data": [
    "/xyz/openbmc_project/network/config",
    "/xyz/openbmc_project/network/config/dhcp",
    "/xyz/openbmc_project/network/eth0",
    "/xyz/openbmc_project/network/eth0/ipv4",
    "/xyz/openbmc_project/network/eth0/ipv4/7d3d3b69",
    "/xyz/openbmc_project/network/eth0/ipv4/abcbdc51",
    "/xyz/openbmc_project/network/eth0/ipv6",
    "/xyz/openbmc_project/network/eth0/ipv6/c67fafda",
    "/xyz/openbmc_project/network/eth1",
    "/xyz/openbmc_project/network/eth1/ipv4",
    "/xyz/openbmc_project/network/eth1/ipv4/7aaae888",
    "/xyz/openbmc_project/network/eth1/ipv4/92df930b",
    "/xyz/openbmc_project/network/eth1/ipv6",
    "/xyz/openbmc_project/network/eth1/ipv6/b0530ca9",
    "/xyz/openbmc_project/network/host0",
    "/xyz/openbmc_project/network/host0/intf",
    "/xyz/openbmc_project/network/host0/intf/addr",
    "/xyz/openbmc_project/network/sit0",
    "/xyz/openbmc_project/network/snmp",
    "/xyz/openbmc_project/network/snmp/manager"
  ],
  "message": "200 OK",
  "status": "ok"
}
```

- Adding `/enumerate` instead of `/list` will also include the attributes of the listed objects.

```
$ curl -k https://$bmc}/xyz/openbmc_project/time/enumerate
{
  "data": {
    "/xyz/openbmc_project/time/bmc": {
      "Elapsed": 1570524502358947
    },
    "/xyz/openbmc_project/time/host": {
      "Elapsed": 1570524502361375
    },
    "/xyz/openbmc_project/time/owner": {
      "TimeOwner": "xyz.openbmc_project.Time.Owner.Owners.BMC"
    },
    "/xyz/openbmc_project/time/sync_method": {
      "TimeSyncMethod": "xyz.openbmc_project.Time.Synchronization.Method.NTP"
    }
  },
  "message": "200 OK",
  "status": "ok"
}
```

## HTTP PUT operations

PUT operations are for updating an existing resource (an object or property), or for creating a new resource when the client already knows where to put it. These require a json formatted payload.

To make curl use the correct content type header use the `-d` option to specify that we're sending JSON data:

```
curl -k -X PUT -d <json> <url>
```

For example, to power on host by doing a PUT:

```
curl -k -X PUT \
  -d '{"data": "xyz.openbmc_project.State.Host.Transition.On"}' \
  https://$bmc}/xyz/openbmc_project/state/host0/attr/RequestedHostTransition
```

## HTTP POST operations

POST operations are for calling methods.

To invoke a method with parameters, for example, Downloading a Tar image via TFTP:

```
curl -k -X POST -d '{"data": ["<Image Tarball>", "<TFTP Server>"]}' \
  https://$bmc}/xyz/openbmc_project/software/action/DownloadViaTFTP
```

To invoke a method without parameters, for example, Factory Reset of BMC and Host:

```
curl -k -X POST -d '{"data": []}' \
https://$ {bmc} /xyz/openbmc_project/software/action/Reset
```

## HTTP DELETE operations

DELETE operations are for removing instances. Only D-Bus objects (instances) can be removed.

For example, to delete the event record with ID 1:

```
curl -k -X DELETE https://$ {bmc} /xyz/openbmc_project/logging/entry/1
```

## Uploading images

It is possible to upload software upgrade images (for example to upgrade the BMC or host software) via REST. The content-type should be set to "application/octet-stream".

For example, to upload an image:

```
curl -k -H "Content-Type: application/octet-stream" \
-X POST -T test https://$ {bmc} /upload/image
```

The operation will either return the version id (hash) of the uploaded file on success:

```
{
  "data": "d2302e3c",
  "message": "200 OK",
  "status": "ok"
}
```

or an error message:

```
{
  "data": {
    "description": "Version already exists or failed to be extracted"
  },
  "message": "400 Bad Request",
  "status": "error"
}
```

Git repository management for enterprise teams powered by Atlassian Bitbucket

Atlassian Bitbucket v5.3.0 · [Documentation](#) · [Contact Support](#) · [Request a feature](#) · [About](#) · [Contact Atlassian](#)