HighPoint Technologies, Inc.

SSD62xx Series (Marvell NevoX RAID Solution)
CLI User Guide

Table of Contents

1 MNV_CLI Software Utility User Guide

This document contains the following sections:

- Introduction
- CLI Features
- Starting the CLI Utility
- Command Conventions and Built-in Help

How to send command to specific adapter via non-shell mode

Non-shell mode

Command

```
mnv_cli [<NVMe Controller ID>] <COMMAND> <OPTIONS>
```

Default NVMe Controller ID is 0.

Example	Description
mnv_cli 1 info -o pd	Show all PD information of NVMe Controller ID 1

- Support Commands and Operations

1.1 Introduction

Marvell 88NR2241 mnv_cli utility Command Line Utility provides a user interface to manage the 88NR2241 (88NR2241) NVMe controller. This document lists the formats and commands of the CLI utility.

1.2 CLI Features

- Switch adapter (controller)
- View adapter, virtual disk and physical disk information
- Create a virtual disk (RAID only)
- Delete a virtual disk (RAID only)
- Namespace attachment
- View SMART of physical disk
- Set, get configuration of adapter and virtual disk (RAID only)
- Flash adapter or physical firmware image

- Rebuild and initialize a virtual disk (RAID only)
- Run media patrol on a virtual disk (RAID only)
- Read event
- Import virtual disk
- Import PD (JBOF only)
- Dump UART information
- Simple API command {VD create/delete, GPIO, Dump SPI, VPD target switch, dump HBA info and GPIO setting for HBA info.}
- Show CLI/API version
- Passthrough NVMe admin command to back-end device
- MI VPD read/write (Linux only)
- OEM data read/write

1.3 Starting the CLI Utility

1.3.1 Starting Linux MNV_CLI

User shall add executable permission for `mnv_cli > chmod 777 mnv_cli`

To enter into CLI prompt: `shell > sudo ./mnv_cli`

1.3.2 Starting Windows MNV_CLI

Run // double click `mnv_cli.exe` program. A Windows CMD Windows starts.

Limitations:

- In Windows, if the 88NR2241 adapter does not have a functional VD presented to the OS, the CLI and API cannot find the adapter. This is a Windows inbox driver limitation.
- Windows driver does not support NVMe notification and hot-plug. Hence, any newly created VD by CLI utility is not detected by Windows until the OS is rebooted (alternatively, the user can try disable/enable 88NR2241 NVMe controller and driver to force a device re-scan).
- Ns (Name Space) admin is not supported by Windows drivers.
- If current user not administrator, please run `mnv_cli.exe` with administrator permission.

1.4 Command Conventions and Built-in Help

This command line utility now can be used in both Linux and Windows environment.

When the utility is used in the OS shell, the syntax is:

```
mnv_cli [adapter id] command [<parameter …>] [-output <file>]
```

`-output` redirects output to the file.

When the utility is used in shell mode, the syntax is:

```
command [<parameter …>]
```

1.4.1 Usage

- `mnv_cli ?:` provides help on all available command line options of the utility.
- `mnv_cli help [command]:` provides detailed help on the command.

1.4.2 Conventions and Built-in Help

When users start `mnv_cli`, the CLI version and API version are displayed. Or user can run `mnv_cli version` to get version information.

Example

CLI Version: 1.0.0.1031

API Version 1.0.0.1031

Support Firmware Version: 1.0.0.1027 above

Help

Legends

`[options]`	the options within [] are optional.
`[<x| y| z>]`	choose none or one of the x, y or z.
`<x| y| z>`	choose one of the x, y or z.

Abbreviations

`VD`	Virtual Disk
`PD`	Physical Disk
`BGA`	Background Activity

Type `'-output [filename]'` to output to a file.

Type `'help'` to display this page.

Type `'help command'` to display the help page of `'command'`.

Type `'command -h'` to display help for `'command'`.

Command name is not case-sensitive and may be abbreviated if the abbreviation is unique. Most commands support both short (-) and long (--) options. Long option names may be abbreviated if the abbreviation is unique. A long option may take a parameter of the form, `--arg=param` or `--arg param`. The option name is case-sensitive, but the option parameter is not.

Command Brief Description

`?`	Get brief help for all commands.
`help`	Get brief help for all commands or detail help for one command.
`create`	Create virtual disk.
`delete`	Delete virtual disk.

get	Get configuration information of NVMe Controller or VD.
set	Set configuration parameters of VD or NVMe Controller.
smart	Get SSD adapter smart info.
flash	Update flash image.
adapter	Default NVMe Controller the following CLI commands refers to.
version	Show API and CLI version
passthru	Passthrough NVMe admin command to back-end device
spi	Read and write SPI data. Maximal size of SPI data is 512 bytes.
vd	Simple API for VD create and delete
log	Simple API for show/on/off SPI log
reset	Simple API for reset back-end device
Note: <Default GPIO setting> Set power: Power pin id 0 means control GPIO4 Power pin id 1 means control GPIO5 Set perstn PD id 0 means control GPIO8 PD id 1 means control GPIO9 PD id 2 means control GPIO10 PD id 3 means control GPIO11	
led	Simple API for control LED of device
vpd	Simple API for switch VPD access target
info	Simple API for display adapter (HBA), virtual disk (VD), physical disk (PD) or Namespace (NS) information.
event	Simple API for getting event from NVMe Controller.
import	Simple API for import VD
init	Simple API for start or stop VD initialization.
mp	Simple API for start or stop media patrol for a virtual disk.
rebuild	Simple API for start or stop VD rebuild.
ns	Simple API for namespace management
dumphba	Simple API for dump HBA info data structure
gpio	Simple API for GPIO setting in HBA info (Only for GPIO 4 to 7)
debuglv	Simple API for switch debug level target
oemdata	Simple API for write OEM data.
vpdread	Read VPD via standard MI-Send and MI-Receive
vpdwrite	Write VPD via standard MI-Send

1.4.3 How to send command to specific adapter via non-shell mode

Non-shell mode

Command

```
mnv_cli [<NVMe Controller ID>] <COMMAND> <OPTIONS>
```

Default NVMe Controller ID is 0.

Example	Description
`mnv_cli 1 info -o pd`	Show all PD information of NVMe Controller ID 1

1.5 Support Commands and Operations

1.5.1 adapter

Objective

This command switches NVMe controller index or lists total NVMe controllers and current default NVMe controller ID in shell mode.

Command

```
adapter [-i <NVMe Controller ID> | -s <bus slot> | --list]
```

Parameters

`-i`	`--id <NVMe Controller ID>`	ID of the NVMe Controller to be selected. If `-i` is not specified, display current NVMe Controller ID
`-s`	`--slot <bus slot>`	The value format is `bus:slot.func`. Refer to lspci in Linux OS. This parameter only supports Linux OS now.
`--list`		Show current NVMe Controller ID and total NVMe Controllers.
`-h`	`--help`	

Return

`0:`	success

Example

`adapter --list`	Show current NVMe Controller ID and total NVMe Controllers.
`adapter -i 1`	Select NVMe Controller 1 to be the default NVMe Controller for the following CLI commands.
`adapter -s 03:00.0`	Select NVMe Controller whose slot is 03:00.0 to be the default NVMe Controller for the following CLI commands.

1.5.2 create

Objective

This command creates a new virtual disk from a set of physical disks. The 88NR2241 is designed to create a maximum of four VDs and four namespaces. Only the VD with namespace attached can be seen from the OS. There is only one VD controller that has a namespace operation (such as, create/delete namespace).

Note: Not supported for JBOF mode.

Command

`create -d <physical disk id list> -r <0|1|10|JBOD> [-n <name>][-i <quick|none>] [-b <128|256|512|1024>] [-h]`

It makes a new virtual disk from a set of physical disks.

Parameters

`-r`	`--raid mode <0|1|10|JBOD> RAID level`	Now supports only RAID 0, RAID 1, RAID 10 and JBOD. The free PD cannot be seen as the OS disk until it is created as JBOD. The first name space of JBOD disk can be seen in OS.
`-d`	`--id <physical disk id list>`	Physical disk IDs used to create the VD, separated by commas.
`-n`	`--name <virtual disk name> Name of virtual disk`	If not set, default is `VD_{VD ID}`. `VD ID`: start from 0. If in the first detect, the `VD ID` does not exist, it is used (for example: If `VD ID 0` does not exist, the name will be `VD_0`). If the blank character is in the name, `<virtual disk name>` must be quoted.
`-i`	`--init <quick|none>`	Initialization mode `(DEFAULT:quick)`. Quick: The first 64KB space of the VD are filled with zero.
`-b`	`--strip block size <128|256|512|1024>`	`(DEFAULT:128)` stripe block size in KB unit.
`-h`	`--help`	

Return

0:	success
1:	failure
157:	need reboot to take effect

Example

create -r 0 -d 0,1 -n "My vd" **Note:** Provide the namespace size based on GB. For example: The available size of VD for creating namespace: 2048 GB. Type namespace size list: For example: Create two namespaces: 20GB and 30GB 20,30 Create two namespace and last one use all remaining size of VD. 20,0 Create one namespace and use all of VD 0 Or type/enter directly without any character.	It creates a RAID 0 VD named 'My VD', using physical disk 0, 1.
create -r JBOD -d 0	Create a JBOD VD using disk 0
create -r 1 -d 0,1	Create a RAID 1 VD using disk 0, 1
create -r 10 -d 0,1,2,3	Create a RAID 10 VD using disk 0, 1, 2, 3
create -r 1 -d 0,1 -b 512	Create a RAID 1 VD using disk 0, 1 and strip block size is 512 KB
create -r 1 -d 0,1 -i none	Create a RAID 1 VD using disk 0, 1 and do not quick initialize

1.5.3 delete

Objective

This command deletes a virtual disk.

Note: Not supported for JBOF mode.

Command

```
delete -i <id> [--waiveconfirmation] [-h]
```

Parameters

-i	--id <id>	ID of LD will be deleted.
-h	--help	

Return

0:	success
1:	failure
157:	need reboot to take effect

Example

delete -i 0	Delete virtual disk with ID equal to 0.

1.5.4 flash

Objective

This command updates the current firmware image.

Note: CLI only supports a file size up to 8 MB (current SPI chip supported size), the file structure checking depends on the firmware version.

Command

```
flash -o <hba [-t type]|pd> [-i <ID>] -f <file> [-s <firmware slot>]
[-c <commit action>] [-b <boot partition id>][-q firmware] [-h]
```

Parameters

-o	--object <hba|pd>	The name of the object to update the firmware. hba - NVMe Controller, default pd - Physical Disk
-t	--type <raw>	Update raw image with specific items. raw - Update all items of raw image.
-i	--id	NVMe Controller ID: to update NVMe Controller firmware. If not specified, update the current NVMe Controller SSD adapter ID of PD info command. **Note**: *To update the disk firmware, this value is necessary.*
-c	--ca <commit action>	The downloaded image replaces the image specified by the Firmware Slot field. The image is activated at the next reset. Windows only accepts 1 and 2. The 88NR2241-Default value is 1.
-f	--file <file name and path>	
-s	--fs <firmware slot>	If the slot is zero, then the controller chooses the firmware slot (Max slot: 7).

Parameters

-o	--object <hba|pd>	The name of the object to update the firmware. hba - NVMe Controller, default pd - Physical Disk
-b	--bp_id <boot partition id>	
-q	--query	firmware, query slot information of NVMe controller or PD.
-h	--help	

Return

0:	success
1:	failure

Example

flash -o hba -f c:\firmware.bin -c 1 -s 1	Update firmware image c:\firmware.bin with commit action as 1 and firmware slot 1 from current NVMe Controller.
flash -o pd -i 0 -f c:\firmware.bin -s 1 -c 1	Update firmware image c:\firmware.bin to back-end device ID 0 with firmware slot 1 and commit action as 1.
flash -o hba -i 0 -t raw -f c:\raw.bin -c 1	Update whole raw image c:\raw.bin to NVMe Controller ID 0.
flash -o hba -i 0 -q firmware	Show slot count, active slot and the next active slot of NVMe Controller ID 0.
flash -o pd -i 0 -q firmware	Show slot count, active slot and the next active slot of PD ID 0 in the current NVMe Controller.

1.5.5 smart

Objective

This command gets SMART info of the back-end device.

Command

```
smart -i <SSD adapter id | NVMe Controller ID <-o hba> > [-h]
```

Parameters

-i	--SSD adapter id or NVMe Controller ID. SSD adapter id, field in PD info. NVMe Controller id.
-o	--object hba, it means getting SMART from adapter

Return

0:	success

Example

smart -i 0	get SSD adapter ID 0 smart info.
smart -i 0 -o hba	get NVMe Controller ID 0 smart info

Parameters

1.5.6 spi

Objective

This command reads/writes SPI data. The maximum size of SPI data is 4096 bytes.

Command

```
spi<-t[read|write]>[--outputfile<outputfile>][--inputfile<inputfile>][-h]
```

Parameters

-t	--type	read: output the SPI data. write: write the SPI data.
	--outputfile<outputfile>	Output data into output file.
	--inputfile<inputfile>	Input data from input file
-h	--help	

Examples

spi -t read	Output raw data on the screen.
spi -t read --outputfile=raw_data	Output raw data into file raw_data.
spi -t write --inputfile=raw_data	Write the SPI of the input file raw_data via the vendor specific command.

1.5.7 set

Objective

This command sets the configuration parameters of VD or NVMe Controller.

Command

Set -o hba -i <NVMe Controller id> [-a on | off] [-r < high|medium|low >]

Set -o vd -i <VD id> [-n <new name>] [-h]

Parameters

-o	--object <hba|vd>	Configure NVMe Controller or VD.
-i	--id <NVMe Controller id | VD id>	HBA ID. Virtual Disk ID.
-n	--name <new name>	If the blank character is in the name, <new name> must be quoted.
-a	--autorebuild <on | off>	
-r	--bgarate <high|medium|low>	
-h	--help	

Return

0:	success
1:	failure

Example

set -o hba -i 0 -a on	Enable auto-rebuild on NVMe Controller 0.
set -o hba -i 0 -r high	Update BGA rate as high on NVMe Controller 0.
set -o vd -i 0 -n my_vd_name	Update name on VD 0.

1.5.8 get

Objective

This command gets the configuration information of VD and NVMe Controller.

Command

```
get -o <hba|vd> -i [id] [-h]
```

Parameters

-o	--object <hba|vd>	Get information of NVMe Controller or VD.
-i	--id <NVMe Controller id|VD id>	NVMe Controller ID or Virtual disk ID. If not specified, all instances of the NVMe Controller or VD will be retrieved.
-h	--help	

Return

0:	success
1:	failure

Example

get -o hba -i 0	Get configuration information of adapter which adapter ID is 0.
get -o vd -i 0	Get configuration information of current adapter which VD id is 0.

1.5.9 vd

Objective

This command is a Simple API feature to create/delete VD via Simple API. Once the Simple API SDK is complete, this feature will be removed from the CLI tool.

Command

```
vd <-a create <-r 0|1|10|JBOD><-d disks> [-b <128|256|512|1024>] [-n
<number of namespaces>] | delete [-i vd id]>[-h]
```

Return

-i	--vd_id	
-a	--action create and delete VD	create: create VD with specific RAID mode and strip block size. delete: delete VD with specific VD ID.
-r	--raid mode <0|1|10|JBOD> RAID level	Now only supports RAID 0, RAID 1, RAID 10 and JBOD. The free PD cannot be seen as OS disk until it is created as JBOD. The first name space of JBOD disk can be seen in OS.
-d	--disk	Number of free disks.
-b	--stripe block size <128|256|512|1024>	(DEFAULT:128) stripe block size in KB unit.
-n	--namespace	Number of namespaces needed to create. If not specific, it creates one namespace. The size of the namespaces is the average of the namespace count in VD.
-h	--help	

Example

create -r 0 -d 0,1 -n "My VD"	Create a RAID 0 VD named 'My VD' using disk 0, 1
create -r JBOD -d 0	Create a JBOD using disk 0
create -r 1 -d 0,1	Create a RAID 1 using disk 0, 2
create -r 10 -d 0,1,2,3	Create a RAID 10 using disk 0, 1, 2, 3
create -r 1 -d 0,1 -b 512	Create a RAID 1 using disk 0, 1 and strip block size is 512 KB
create -r 1 -d 0, 1 -i none	Create a RAID 1 using disk 0, 1 and do not quick initialize

1.5.10 log

Objective

This command is a Simple API feature to create debug log via Simple API. The function works as a dump.

Command

```
log <-a show [--outputfile=log.txt] | on | off > [-h]
```

Parameters

-a	--action show and set on/off	show: show SPI log on: enable SPI log off: disable SPI log
	--outputfile <outputfile>	Output the data into the output file.
-h	--help	

Example

log -a show	Show SPI log.
log -a show --outputfile=log.txt	Show SPI log into the text file named log.txt
log -a on	Enable SPI log.
log -a off	Disable SPI log.

1.5.11 reset

Objective

This command is a Simple API feature to reset the back-end device. Once the Simple API SDK is complete, this feature is removed from the CLI tool.

Command

```
reset -i <pdid> -t <device | perstn | power -a <cycle|on|off> > [-h]
```

Note:

<Default GPIO setting> Set power: pdid 0 means control GPIO4 pdid 1 means control GPIO5
Set perstn: pdid 0 means control GPIO8 pdid 1 means control GPIO9 pdid 2 means control GPIO10 pdid 3 means control GPIO11

-i	--pdid	
-t	--type	device: devbitmap and power device, device disable and enable power: power off and on perstn: PCIe reset cycle
-a	--action	power: cycle on and off only cycle: for power off and on a cycle (Default) on: power on off: power off
-h	--help	

Example

`reset -i 0 -t device`	Device PD ID 0 as reset cycle.
`reset -i 1 -t perstn`	Device PD ID 1 as PCIe cycle reset.
`reset -i 1 -t power -a cycle`	Device PD ID 1 as a power cycle reset.
`reset -i 1 -t power -a on`	Device PD ID 1 as power on.
`reset -i 1 -t power -a off`	Device PD ID 1 as power off.

1.5.12 led

Command

```
led -o <ep|rc> -i <ep id | rc id> -a <on|off|sb|qb >  [-h]
```

Objective

This is a Simple API feature to locate NVMe Controller (HBA) and PD via Simple API.

Parameters

`-o`	`--object, endpoint or root complex`	
`-i`	`--back-end device id`	
`-a`	`--action, off, on, sb(slow blink) and qb(quick blink)`	
`-h`	`--help`	

Example

`led -o ep -i 0 -a on`	Setting end point 0 of 88NR2241 as status on.
`led -o ep -i 0 -a off`	Setting end point 0 of 88NR2241 as status off.
`led -o rc -i 1 -a sb`	Setting root complex 1 of 88NR2241 as status slow blink.
`led -o rc -i 2 -a qb`	Setting root complex 2 of 88NR2241as status quick blink.

1.5.13 vpd

Objective

This command is a Simple API feature to switch VPD access target via Simple API.

Command

```
vpd <-t eeprom|spi > [-h]
```

-t	--target	Set access EEPROM or SPI. eeprom, set access target to EEPROM spi, set access target to SPI
-h	--help	

Example

vpd -t eeprom	Set access target to EEPROM.
vpd -t spi	Set access target to SPI.

1.5.14 ns

Objective

This is a Simple API feature to switch VPD access target via Simple API. It is used for namespace management.

Command

```
ns -p | -a <init -i <pdid> -n <number of namespace> | share -s
[yes|no] [-i <ns id> | -e <eui64>] | create -i <pdid|vdid> -c
<ctrlid> -l<LBA size> | [attach|detach] [-i <ns id> | -e <eui64>] -d
<ctrl id list> > [-h]
```

Parameters

-p	--primary <ctrl id>	Show primary controller ID.
-a	--actions init, share, create, attach and detach	init: initial namespace for specific instance (JBOF only) share: setting NMIC of namespace for private or sharing (JBOF only) create: create namespace (JBOF only) attach: attach namespace detach: detach namespace
-i	--id <pd id | ns id | vd id>	For init action, need input one PD ID. For share action, need input one namespace ID. For create action, need input one PD/VD id.
-e	--EUI64 <ns unique identifier	Notice: only support attached namespace For share action, need input one EUI64 identifier id or namespace id

-n	--nn <number of namespace>	For init action, need the input number of namespace.
-s	--sharing, for action 'share'	no, only current controller can access namespace yes, allow all controller can access namespace
-c	--ctrllist controller id, for action 'create' and 'share'	0: create shared namespace 1~38: controller identifiers 1~6: means physical function (PF) 0 through 5 respectively. 7~38: denotes virtual function (VF) 0 through 31.
-l	--lba	LBA size
-d	--id <controller ids>	For attach and detach action, input multiple controller ids, use \',\' to separate
-h	--help	

Example

ns -a init -i 0 -n 1	init pd ID 0 with 1 namespace.
ns -a share -i 1 -s yes	set NMIC of namespace ID 1 controller ID 0 as sharing.
ns -a share -e 00504325385b7601 -s yes	set NMIC of namespace EUI64 id 00504325385b7601 as sharing.
ns -a share -i 1 -s no -c 1	set NMIC of namespace ID 1 controller ID 0 as private.
ns -a share -e 00504325385b7601 -s no -c 1	set NMIC of namespace EUI64 id 00504325385b7601 controller id 0 as private.
ns -a create -i 0 -c 1 -l 0x2800000	create 20GB ns of disk id 0 PF controller id 0 which LBA is 0x2800000.
ns -a create -i 0 -c 7 -l 0x5000000	create 40GB ns of disk id 0 VF controller id 0 which LBA is 0x5000000
ns -a attach -i 1 -d 0	attach namespace ID 1 to controller 0.
ns -a detach -i 1 -d 0,1	detach namespace ID 1 from controller 0 and 1
ns -a detach -e 00504325385b7601 -d 0,1	detach namespace EUI64 id 00504325385b7601 from controller 0 and 1
ns -p ctrl	show current primary controller ID.
ns --list	show current namespace information.

1.5.15 dumphba

Objective

This is a Simple API feature to switch VPD access target via Simple API. It is used to dump hba info data structure.

Command

```
dumphba [--outputfile <outputfile>] [-a show] [-h]
```

Parameters

	`--outputfile <outputfile>`	Output data into the output file.
`-a`	`--show`	Show parts value of HBA info readable.
`-h`	`--help`	

Example

`Dumphba`	Output raw data
`dumphba --outputfile=raw_data`	Output raw data into file raw_data.
`dumphba -a show`	Output parts value of HBA info.

1.5.16 gpio

Objective

This command is a Simple API feature to set GPIO via Simple API (GPIO setting in HBA info [only for GPIO 4 to 7]).

Command

```
gpio --func-switch=<func-switch bitmap> --level=<level bitmap> --
value=<value bitmap> --set-default=<yes|no> [-h]
```

Parameters

	`--func`	switch, bitmaps for switch to GPIO or other function. 0: Other function 0x30: GPIO4, 5 0xC0: GPIO6, 7 0xF0: GPIO4, 5, 6, 7 Only supports the above settings.
	`--level`	bitmaps for the default voltage level base on the customer's schematic design. 0: low 1: high

	`--value`	bitmaps for the GPIO default value (default: 0x30, it means GPIO4, 5)
	`--set`	default, whether want to use the setting from HBA info. 0: no 1: yes
`-h`	`--help`	

Example

`gpio --set-default=no`	use firmware default setting.
`gpio --set-default=yes`	use customer-defined setting that is stored in HBA info.
`gpio --func-switch 0x30`	use GPIO function in GPIO4, 5 (bitmap: bit4, 5).
`gpio --func-switch 0x00`	use other function in GPIO4, 5 (bitmap: bit4, 5).
`gpio --func-switch 0x30 --level=0x10 --value=0x10 --set-default=yes`	set value '1' to GPIO4, 5 and GPIO4 pin level is high.
`gpio --func-switch 0x30 --level=0x10 --value=0x00 --set-default=yes`	set value '0' to GPIO4, 5 and GPIO4 pin level is high.
`gpio --func-switch 0x30 --level=0x00 --value=0x10 --set-default=yes`	set value '1' to GPIO4, 5 and GPIO4 pin level is low.
`gpio --func-switch 0x30 --level=0x00 --value=0x00 --set-default=yes`	set value '0' to GPIO4, 5 and GPIO4 pin level is low.
`gpio --func-switch 0x30 --level=0x30 --value=0x30 --set-default=yes`	set value '1' to GPIO4, 5 and GPIO4, 5 pin level is high.
`gpio --func-switch 0x30 --level=0x30 --value=0x00 --set-default=yes`	set value '0' to GPIO4, 5 and GPIO4, 5 pin level is high.
`gpio --func-switch 0x30 --level=0x00 --value=0x30 --set-default=yes`	set value '1' to GPIO4, 5 and GPIO4, 5 pin level is low.
`gpio --func-switch 0x30 --level=0x00 --value=0x00 --set-default=yes`	set value '0' to GPIO4, 5 and GPIO4, 5 pin level is low.

1.5.17 event

Objective

To get the newest event from the adapter.

Command

```
event [-c event count][-h]
```

Parameters

`-c`	`--count <event count>`	`If count is zero, it will show all events` `If count is not zero, it will show latest specific events.`
`-h`	`--help`	

Return

`0:`	success

Example

`Event -c 0`	Get events from current NVMe Controller. For the current design, it displays all the event pages. It is 4k/per page and the total number of pages is 16.
`event -c 20`	Get latest 20 events from current NVMe Controller.

1.5.18 import

Objective

This command is a Simple API feature to import a VD when an importable VD roams from one controller to another. If NVMe Controller supports the RAID mode, this function will import VD.

If VD is created from one controller and the SSD with VD roamed to another controller, the VD needs to be imported first before the firmware reports the VD to the OS.

Note: The user can use `> info -o VD` command to check if the VD status to be imported or not.

Command

```
import [-i <NVMe Controller ID>] -l <VD ID>
```

Parameters

`-i`	`--id <NVMe Controller ID>`	Select ID of the NVMe Controller. If -i is not specified, display current NVMe Controller ID.
`-l`	`--ldid <VD ID>`	ID of the VD to be selected.
`-h`	`--help`	

Example

`import -i 0 -l 0`	Import VD ID 0 into NVMe Controller 0.

1.5.19 info

Objective

This command is a Simple API feature to display NVMe controllers/physical disks/virtual disks info. It dumps the information of NVMe controller, virtual disks or physical disks.

Command

```
info -o <hba|vd|pd|ns [-t <host|vd> [-v <vd_id>]] > [-i <id>] [-h]
```

Parameters

-o	--object <hba|vd|pd|ns>	hba: NVMe Controller vd: Virtual Disk pd: Physical Disk ns: Name Space
-i	--id <HBA id|VD id|PD id|NS id>	HBA ID, Virtual disk ID, Physical disk ID or Namespace ID. If not specified, all instances of the HBA, VD, PD or NS will be retrieved.
-t	--type <Host | VD>	For ns info only. Host, namespace information which report for OS (default) VD, namespace information which can be listed from specific VD ID. If not specified, the default type is Host. CLI will list all namespace information which report for OS.
-v	--vd_id <VD id>	For ns info of VD only. This is listed for all the namespace information of specific VD ID. If not specified, all instances of the NS will be retrieved.
-h	--help	

Return

0:	success
1:	failure

Example

info -o hba -i 1	Information of NVMe controller ID 1.
info -o vd -i 1	Information of VD ID 1.
info -o pd -i 1	Information of PD ID 1.
info -o ns	All namespace information that NVMe controller reports to OS.
info -o ns -t vd -v 1	All namespace information that VD ID 1 contains.

1.5.20 init

Objective

This command is a Simple API feature to start or stop VD initialization.

Note:

Command

```
init [-a <start|stop>] -i <id> [-h]
```

Note: For now, only provides VD full initialization. The VD must be functional. It does not provide support for JBOF mode.

Parameters

-a	--action <start|stop>	(DEFAULT:start) - Initialization action to be performed on VD.
-i	--id <id>	VD ID.
-h	--help	

Return

0:	success
1:	failure

Example

init -a start -i 0	Start full background initialization on VD 0.
init -a stop -i 0	Stop full background initialization on VD 0.

1.5.21 rebuild

Objective

This command is a Simple API feature to start or stop VD rebuild on refer PD.

Command

```
rebuild [-a <start|stop>] -i <vd id> -d <pd id> [-h]
```

Note: Does not support JBOF mode.

Parameters

-a	--action <start|stop>	(DEFAULT:start) - Rebuild action to be performed on VD.
-i	--id <vd id>	VD ID.
-d	--pdid <id>	PD ID.
-h	--help	

Return

0:	success
1:	failure

Example

rebuild -a start -i 0 -d 0	Start background rebuild on VD 0 with PD 0.
rebuild -a stop -i 0	Stop background rebuild on VD 0.

Return

1.5.22 mp

Objective

This command is a Simple API feature to start or stop media patrol for a virtual disk.

Command

```
mp [-a <start|stop >] -i <VDid> [-h]
```

Note: This feature only works with VD whose RAID is RAID1 or RAID10 and functional. When the status is paused, it cannot abort until the status is running. Does not support JBOF mode.

Parameters

-a	--action <start|stop>	(DEFAULT:start) - media patrol action to be performed on a VD. Only RAID1 and RAID10 support media patrol. If VD is not functional, media patrol may fail.
-i	--vdid <VDid>	VD ID.
-h	--help	

Return

0:	success
1:	failure

Example

mp -a start -i 0	Start media patrol on VD 0.
mp -a stop -i 0	Stop media patrol on VD 0.

1.5.23 oemdata

Objective

This command is a Simple API feature that writes OEM data to the NVMe controller.

Command

```
oemdata [-v <SubVendor ID/SVID> -d <SubDevice ID/SDID> -s <Serial
Number> -m <Model Number>] [-h]
```

Parameters

-v	--vid <SubVendor ID/SVID>
-d	--did <SubDevice ID/SDID>
-s	--sn <Serial Number>
-m	--mode <Model Number>
-h	--help

Return

0:	success
1:	failure

Example

`oemdata -v 0x11ab -d 0x2243 -s serialnumber -m model`	Write Subvendor ID 0x11ab, SubDevice ID 0x2243, serial number 'serialnumber' and model number 'model' into OEM data.

1.5.24 passthru

Objective

This command is a pass-through NVMe admin command to the back-end device.

Command

```
passthru -t <admin> -r <read | write> -i <SSD adapter id>
         [-o <opcode>]
         [-n <ns id>][--cdw2=<cdw2>][--cdw3=<cdw3>]
         [--cdw10=<cdw10>][--cdw11=<cdw11>][--cdw12=<cdw12>]
         [--cdw13=<cdw13>]
         [-d <data-len>]
        [--outputfile <outfile>][--inputfile <inputfile>]
        [--showcmd][-h]
```

Note: CDW14 and CDW15 are used by passthru command.

Parameters

`-t`	`--type <admin>`	Default: admin. For now we only support passthru NVMe admin command set to backend device.
`-r`	`--req <read | write | none>`	Send read and write command. Non- direct command can use read or write command to send.
`-i`	`--id <SSD adapter id>`	This will map to the SSD adapter ID of the PD info command.
`-o`	`--opcode <opcode>`	This is the opcode of the standard NVMe admin command set. Accept value by decimal or hexadecimal.

-n	--nsid <ns id>	Bytes 15:12 in admin command format. --cdw10 <cdw10> Bytes 43:40 in admin command format. --cdw11 <cdw11> Bytes 47:44 in admin command format. --cdw12 <cdw12> Bytes 51:48 in admin command format. --cdw13 <cdw13> Bytes 55:52 in admin command format.
-d	--data-len <data-len>	Set the data length buffer for admin command.
	--outputfile <outputfile>	Output data into the output file.
	--inputfile <inputfile>	Input data from the input file.
-h	--get help of passthru command help	

Example

passthru -i 0 -o 0xa -n 1 --cdw10=0x103 --data-len=4096 -r read	Send a read command for default LBA Range Type to get feature to SSD adapter ID 0 nsid 1 of 88NR2241.
passthru -i 0 -o 0x2 -n 0xffffffff --cdw10=0x7f0002 --data-len=512 -r read	Send a read command for getting SMART information SSD adapter id 0 of 88NR2241
passthru -i 0 -o 6 --cdw10=1 --data-len=4096 -r read	Send a read command for getting the primary controller information to identify the SSD adapter ID 0 of 88NR2241.
passthru -i 1 -o 0x6 --cdw10=1 --data-len=4096 -r read --outputfile id_data	Send a read command for getting the primary controller information to identify the SSD adapter ID 1 of 88NR2241 into output file "id_data"
passthru -i 1 -o 0x15 --cdw10=0 --data-len=4096 -n 1 --inputfile prp_data -r write	Send a write command for the SSD adapter ID 1 to attach namespace 1 with the controller ID list of the input file "prp_data".

1.5.25 vpdread

Objective

This command is to read the VPD via standard MI-Send and MI-Receive.

Command

```
vpdread -o <offset> -l <data length> [-f <outputfile>] [-h]
```

Note: Does not support Windows.

-o	--offset	Offset of the VPD data.
-l	--datalen	Data length of the VPD data from offset to get.
-f	--outputfile <outputfile>	Output data into the output file.
-h	--help	

Example

vpdread -o 0 -l 32	Read the VPD data from offset 0. The data length is 32 bytes.
vpdread -o 0 -l 32 -f raw_data	Read the VPD data from offset 0 (data length is 32 bytes) and write into the file raw_data.

1.5.26 vpdwrite

Objective

This command is to write the VPD via standard MI-Send.

Command

```
vpdwrite -o <offset> -l <data length> -f <input file> [-h]
```

Note: Does not support Windows.

Parameters

-o	--offset	Offset of the VPD data.
-l	--datalen	Data length of the input data from offset to set.
-f	--inputfile <inputfile>	Input the required data into the VPD.
-h	--help	

Example

vpdwrite -o 0 -l 32 -f raw_data	Write the VPD data from offset 0 (data length is 32 bytes) from the file raw_data.

Thank You