/* JSON-RPC Methods for Mrvl SPDK Application. */

----

mrvl_nvm_init

Start initialization of Mrvl Nvme. This RPC can be called only once and must be called before calling any of the API's below.

Parameters

This method has no parameters.

Response

Returns 0 on success or error code as per Error-codes table1 on error.

Example

Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_init"
}
```

Example response:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
      “status”: 0
      }
}
```

----

mrvl_nvm_get_offload_cap

Get capabilities of the DPU for NVMeOF offload.

Parameters

This method has no parameters.

Response

Json object containing NVMeOF offload capabilities of DPU.

Returns status 0 on success or error code as per Error-codes table1 on error.

Example

Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_get_offload_cap"
}
Example response:

{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
      "status": 0,
      “sdk_version”: “11.22.06”,
      "nvm_version": "1.3",
      "num_pcie_domains": 1,
      "num_pfs_per_domain": 1,
      "num_vfs_per_pf": 16,
      "total_ioq_per_pf": 128,
      "max_ioq_per_pf": 128,
      "max_ioq_per_vf": 128,
      "max_subsystems":16,
      "max_ns_per_subsys": 8,
      "max_ctrlr_per_subsys: 16
  }
}
```

----

mrvl_nvm_get_subsys_count

Parameters
This method has no parameters.

Response
Count of subsystems. Also returns status 0 on success or error code as per Error-codes table1 on error.

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_get_subsys_count"
}
Example response:

{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
      "status": 0,
      "count": 2
  }
}
```
----

mrvl_nvm_get_subsys_list

Parameters
This method has no parameters.

Response
Json object containing list of created subsystems.
Returns status 0 on success or error code as per Error-codes table1 on error.

Example
Example request:

```
        {
          "jsonrpc": "2.0",
          "id": 1,
          "method": "mrvl_nvm_get_subsys_list"
        }
```
Example response:

```
        {
          "jsonrpc": "2.0",
          "id": 1,
          "result": {
              "status":0,
              "subsys_list": [
                    {
                          "subnqn": "nqn.2014-
08.org.nvmexpress.discovery"
                    },
                    {
                          "subnqn": "nqn.2016-06.io.spdk:cnode3"
                    }
              ]
          }
        }
```

----

mrvl_nvm_create_subsystem

Parameters
Subsystem parameters

Name	**Optional**	**Type**	**Description**
subnqn	required	string	Subsystem nqn

mn/sn	required	string	Model number and serial number
Max_namespaces	optional	number	Maximum number of namespaces supported by the subsystem being created. Default 16
Min/max ctrlr_id	Optional	Number	Minimum and maximum controller id's to be used for subsystem. Default Min 0, Max 32

Response
Returns 0 on success or error codes as per the Error-Codes Table1.

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_create_subsystem",
  "params": {
      "subnqn": "nqn.2014-08.org.nvmexpress.discovery",
      "mn": "OCTEON NVME 0.0.1",
      "sn": "OCTNVME0000000000002",
      "max_namespaces": 16,
      "min_ctrlr_id": 1,
      "max_ctrlr_id": 8
  }
}
```

Example response:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": 0
}
```

----

mrvl_nvm_deletesubsystem

Parameters

Name	Optional	Type	Description
subnqn	required	string	Subsystem nqn

Response: Returns 0 on success or error code as per Error-Codes table 1 on error.

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_delete_subsystem",
  "params": {
     "subnqn": "nqn.2014-08.org.nvmexpress.discovery",
  }
}
Example response:

{
  "jsonrpc": "2.0",
  "id": 1,
  "result": 0
}
```

----

mrvl_nvm_deinit

Parameters
This method has no parameters.

Response
Status code 0 on success or error code as per Error-codes table1 on error

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_deinit"
}
Example response:

{
  "jsonrpc": "2.0",
  "id": 1,
  "result": 0
}
```

/* SUBSYSTEM API */

mrvl_nvm_subsys_get_info

Parameters

Name	**Optional**	**Type**	**Description**
subnqn	optional	string	Subsystem nqn. If no nqn provided, info for all subsystems will be returned.
level	optional	string	This can take values “brief” or “detail”. “brief” is the default. “brief” gives basic information about the subsystem and all its objects(ns, controllers). With “detail” option, all namespace details like uuid etc and all controller details like sqes, cqes etc will also be returned.

Response
Subsystem info and status code. On error, it will return error-code as per Error-Codes table1

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_subsys_get_info",
  "params": {
     "subnqn": "nqn.2014-08.org.nvmexpress.discovery",
  }
}
```

Example response:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
     "status": 0,
     "subsys_list": [
           {
           "subnqn": "nqn.2014-08.org.nvmexpress.discovery",
           "mn": "OCTEON NVME 0.0.1",
           "sn": "OCTNVME0000000000002",
           "max_namespaces": 16,
           "min_ctrlr_id": 1,
           "max_ctrlr_id": 8,
           "num_ns": 2,
           "num_total_ctrlr": 2,
           "num_active_ctrlr": 2,
           "ns_list": [
                 {
                      “ns_instance_id”: 1,
                      “bdev”: “bdev01”,
                      "ctrlr_id_list": [
                           {
                                 "ctrlr_id": 1
                           },
                           {
                           "ctrlr_id": 2
                           }
                      ]

                 },
                 {
                      "ns_instance_id": 1,
                      "bdev": "bdev02",
                      "ctrlr_id_list": [
                           {
                           "ctrlr_id": 3
                           }
                      ]

                 }
           ]
      }
      ]

  }
}
```

----

mrvl_nvm_subsys_alloc_ns

Parameters

Name	Optional	Type	Description
subnqn	required	string	Subsystem nqn.
bdev	required	string	Name of bdev
nguid	optional	string	Namespace guid
eui64	optional	string	64 bit unique id
uuid	optional	string	If not provided, bdev's uuid or autogenerated uuid will be provided by application.
Ns_instance_id	optional	Number	If provided, same will be returned in response.
Share_enable	Optional	number	This can take value 0/1. Default is 1.

Response

Namespace instance id and status code. Ns_instance_id returned will be same if given in request, else will be an integer generated by app, unique at subsystem level. This id along with subnqn will be used to uniquely refer to this namespace by other API's.

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_subsys_alloc_ns",
  "params": {
     "subnqn": "nqn.2014-08.org.nvmexpress.discovery",
     "nguid": "0x25f9cbc45d0f976fb9c1a14ff5aed4b0",
     "eui64": "0xa7632f80702e4242",
     "uuid": "0xb35633240b77073b8b4ebda571120dfb",
     “ns_instance_id”: 1
     "share_enable": 1,
     "bdev": "bdev01"
  }
}
```

Example response:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
     "status": 0,
     “ns_instance_id”: 1
  }
}
```

----

mrvl_nvm_subsys_unalloc_ns

Parameters

Name	**Optional**	**Type**	**Description**
subnqn	required	string	Subsystem nqn.
Ns_instance_id	required	number	Namespace object id returned during alloc_ns call.

Response
status code 0 on success or error code as per Error-codes table1.

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_subsys_unalloc_ns",
  "params": {
     "subnqn": "nqn.2014-08.org.nvmexpress.discovery",
    “ns_instance_id”: 1
  }
}
Example response:

{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
      "status": 0,

  }
}
```
----

mrvl_nvm_subsys_get_ns_list

Parameters
List of subsystems' nqn.

Name	Optional	Type	Description
subnqn	required	string	Subsystem nqn.

Response
status code 0 on success or error code as per Error-codes table1. Also returns list of namespaces ns_instance_id underlying bdev name and controllers.

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_subsys_get_ns_list",
  "params": [
      {
      "subnqn": "nqn.2014-08.org.nvmexpress.discovery",
      },
      {
      "subnqn": "nqn.2016-06.io.spdk:cnode1",
      }
      ]
}
```
Example response:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
      "status": 0,
      "ns_list": [
                    {
                          "ns_instance_id":1,

                          “bdev”: “bdev01”,
                          "ctrlr_id_list": [
                                {
                                      "ctrlr_id": 1
                                }
                          ]

                    },
                    {
                          "ns_instance_id": 2,
                          “bdev”: “bdev02”,
                          "ctrlr_id_list": [
                          ]

                    }
              ]
          }
        }
```

----

mrvl_nvm_subsys_create_ctrlr

Parameters
Subsystem nqn, Controller params

Name	**Optional**	**Type**	**Description**
subnqn	required	string	Subsystem nqn.
Pci_segment_id	Optional	Number	Default value 1 will be used
instance_id	Optional	Number	PF/VF id. If not provided next available number will be used. 0 for PF and 1 to 32 for VF.
Max_nsq	Optional	Number	Max submission queues. Default 2
Max_ncq	Optional	Number	Max completion queues. Default 2
sqes	Optional	Number	Max submission queue entry as index of 2. Default 6.
cqes	Optional	Number	Max completion queue entry as index of 2. Default 4.

Response
Ctrlr id and status code 0 on success or error code as per Error-codes table1.

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_subsys_create_ctrlr",
  "params": {
          "subnqn": "nqn.2014-08.org.nvmexpress.discovery"
          "pcie_domain_id": 1,
          "is_pf": 1,
          "instance_id": 0,
          ,
          "max_nsq": 4,
          "max_ncq": 4,
     }
}
Example response:

{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
     "status": 0,
     "ctrlr_id": 1
  }
}
```

----

mrvl_nvm_subsys_update_ctrlr

Parameters
Subsystem nqn, Controller params

Name	**Optional**	**Type**	**Description**
subnqn	required	string	Subsystem nqn.
Ctrlr_id	required	Number	Controller ID to be updated
Max_nsq	Optional	Number	Max submission queues. Default 2
Max_ncq	Optional	Number	Max completion queues. Default 2
sqes	Optional	Number	Max submission queue entry as index of 2. Default 6.
cqes	Optional	Number	Max completion queue entry as index of 2. Default 4.

Response
Ctrlr id and status code 0 on success or error code as per Error-codes table1.

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_subsys_create_ctrlr",
  "params": {
          "subnqn": "nqn.2014-08.org.nvmexpress.discovery"
          "ctrlr_id": 0,
          "max_nsq": 4,
          "max_ncq": 4,
    }
}
Example response:

{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
    "status": 0,
  }
}
```

----

mrvl_nvm_subsys_remove_ctrlr

Parameters
Subsystem nqn, controller ID.

Name	**Optional**	**Type**	**Description**
subnqn	required	string	Subsystem nqn.
Ctrlr_id	required	Number	Controller id

Response
status code 0 on success or error code as per Error-codes table1

Example
Example request:

```
    {
      "jsonrpc": "2.0",
      "id": 1,
      "method": "mrvl_nvm_subsys_remove_ctrlr",
      "params": {
                "subnqn": "nqn.2014-08.org.nvmexpress.discovery",
                "cntlr_id": 1
          }
    }
    Example response:

    {
      "jsonrpc": "2.0",
      "id": 1,
      "result": {
          "status": 0
      }
    }
```

----

mrvl_nvm_subsys_get_ctrlr_list

Parameters

List of subsystems' nqn

Name	Optional	Type	Description
subnqn	required	string	Subsystem nqn.

Response

status code 0 on success or error code as per Error-codes table1, and List of controller IDs.

Example

Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_subsys_get_ctrlr_list",
  "params": [
      {
           "subnqn": "nqn.2014-08.org.nvmexpress.discovery"
      },
      {
           "subnqn": "nqn.2016-06.io.spdk:cnode1"
      }
      ]
}
```

Example response:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
      "status": 0,
      "ctrlr_id_list": [
           {
                "ctrlr_id": 1
           },
           {
                "ctrlr_id": 2
           }
      ]
  }
}
```

/* NS API */

mrvl_nvm_ns_get_stats

Parameters

Name	Optional	Type	Description
subnqn	required	string	Subsystem nqn.
Ns_instance_id	required	number	Namespace object id returned during alloc_ns call.

Response
status code 0 on success or error code as per Error-codes table1 and Namespace statistics

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_ns_get_stats",
  "params": {
      "subnqn": "nqn.2016-06.io.spdk:cnode1",
      “ns_instance_id”: 1
      }
}
```

Example response:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
      "status": 0,
      "num_read_cmds": 0,
      "num_read_bytes": 0,
      "num_write_cmds": 0,
      "num_write_bytes": 0,
      "num_errors": 0,
      “total_read_latency_in_us”:
      “total_write_latency_in_us”:
      “Stats_time_window_in_us”: 10
  }
}
```

---

mrvl_nvm_ns_get_ctrlr_list

Parameters

Name	Optional	Type	Description
subnqn	required	string	Subsystem nqn.
Ns_instance_id	required	number	Namespace object id returned during alloc_ns call.

Response
status code, 0 on success or error codes as per Error-codes table1 and List of controller IDs.

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_ns_get_ctrlr_list",
  "params": [
      "subnqn": "nqn.2016-06.io.spdk:cnode1",
      “ns_instance_id”: 1
}
```

Example response:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
      "status": 0,
      "ctrlr_id_list": [
           {
                "ctrlr_id": 1
           },
           {
                "ctrlr_id": 2
           }
      ]
  }
}
```

----

mrvl_nvm_ns_get_info

Parameters
Subsystem nqn, ns_instance_id.

Name	**Optional**	**Type**	**Description**
subnqn	required	string	Subsystem nqn.
Ns_instance_id	required	number	Namespace object id returned during alloc_ns call.

Response
status code, 0 on success or error code as per Error-codes table1 and Namespace info attributes.

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_ns_get_info",
  "params": {

      "subnqn": "nqn.2016-06.io.spdk:cnode1",
      “ns_instance_id”: 1
      }
}
```

Example response:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
      "status": 0,
      "nguid": "0x25f9cbc45d0f976fb9c1a14ff5aed4b0",
      "eui64": "0xa7632f80702e4242",
      "uuid": "0xb35633240b77073b8b4ebda571120dfb",
      "nmic": 1,
      “bdev”: “bdev01”,
      "num_ctrlrs": 1,
      "ctrlr_id_list": [
            {
                  "ctrlr_id": 1
            }
      ]

  }
}
```

```
-----------------------------------------------------------
/* CTRLR API */
```

mrvl_nvm_ctrlr_attach_ns

Parameters
Subsystem nqn, Controller id, ns_instance_id

Name	Optional	Type	Description
subnqn	required	string	Subsystem nqn.
Ctrl_id	required	String	Controller ID
Ns_instance_id	required	number	Namespace object id returned during alloc_ns call.

Response
status code, 0 on success or status code as mentioned in Error-codes table1.

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_ctrlr_attach_ns",
  "params": {
      "subnqn": "nqn.2016-06.io.spdk:cnode1",
      “ctrl_id”: 1,
      "ns_instance_id": 1
      }
}
```

Example response:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
      "status": 0,
  }
}
```

----

mrvl_nvm_ctrlr_detach_ns

Parameters
Subsystem nqn, Controller id, ns_instance_id.

Name	Optional	Type	Description
subnqn	required	string	Subsystem nqn.
Ctrl_id	required	String	Controller ID
Ns_instance_id	required	number	Namespace object id returned during alloc_ns call.

Response
status code, 0 on success or error code as per Error-codes table1.

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_ctrlr_detach_ns",
  "params": {
      "subnqn": "nqn.2016-06.io.spdk:cnode1",
      "ctrlr_id": 1,
      "ns_instance_id": 1,
      }
}

Example response:

{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
      "status": 0,
  }
}
```

----

mrvl_nvm_ctrlr_get_info

Parameters
Subsystm nqn, Controller ID.

Name	Optional	Type	Description
subnqn	required	string	Subsystem nqn.
Ctrl_id	required	String	Controller ID

Response
status code, 0 on success or error code as per Error-codes table1 and Controller info attributes.

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_ctrlr_get_info",
  "params": {
      "subnqn": "nqn.2016-06.io.spdk:cnode1",
      "ctrlr_id": 1
      }
}
```

Example response:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
      "status": 0,
      "pcie_domain_id": 1,
      "pf_id": 1,
      "vf_id": 1,
      "ctrlr_id": 1,
      "max_nsq": 4,
      "max_ncq": 4,
      "mqes": 2048
      "ieee_oui": "005043",
      "cmic": 6,
      "nn": 16,
      "active_ns_count": 4,
      "active_nsq": 2
      "active_ncq": 2,
      "mdts": 9,
      "sqes": 6,
      "cqes": 4
  }
}
```

----

mrvl_nvm_ctrlr_get_stats

Parameters
Subsystm nqn, Controller ID.

Name	Optional	Type	Description
subnqn	required	string	Subsystem nqn.
Ctrl_id	required	String	Controller ID

Response
status code, 0 on success and Controller stats attributes.

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_ctrlr_get_stats",
  "params": {
      "subnqn": "nqn.2016-06.io.spdk:cnode1",
      "ctrlr_id": 1
      }
}
```

Example response:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
      "status": 0,
      "num_admin_cmds": 0,
      "num_admin_cmd_errors": 0,
      "num_async_events": 0,
      "num_read_cmds": 0,
      "num_read_bytes": 0,
      "num_write_cmds": 0,
      "num_write_bytes": 0,
      "num_errors": 0,
      “total_read_latency_in_us”:
      “total_write_latency_in_us”:
      “Stats_time_window_in_us”: 10

  }
}
```
----

mrvl_nvm_ctrlr_get_ns_stats

Parameters
Subsystm nqn, Controller ID, ns_instance_id

Name	Optional	Type	Description
subnqn	required	string	Subsystem nqn.
Ctrl_id	required	String	Controller ID
Ns_instance_id	required	number	Namespace object id returned during alloc_ns call.

Response
status code, 0 on success or error-codes as per Error-Codes table1 and Controller namespace stats attributes.

Example
Example request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "mrvl_nvm_ctrlr_get_ns_stats",
  "params": {
      "subnqn": "nqn.2016-06.io.spdk:cnode1",
      "ctrlr_id": 1,
      “ns_instance_id”: 1
      }
}
```

Example response:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
      "status": 0,
      "num_read_cmds": 0,
      "num_read_bytes": 0,
      "num_write_cmds": 0,
      "num_write_bytes": 0,
      "num_errors": 0,
      "total_read_latency_in_us": 0,
      "total_write_latency_in_us": 0,
      “stats_time_window_in_us”: 10
  }
}
```

```
-----------------------------------------------------------------------
-----------------------------------------------------
C-API

/*
 * Copyright (C) 2022 Marvell International Ltd.
 *  SPDX-License-Identifier: BSD-3-Clause
 */

#ifndef __MRVL_NVM_API_H__
#define __MRVL_NVM_API_H__

#define NVM_SUBNQN_SIZE 256
#define NVM_MN_SIZE 40
#define NVM_SN_SIZE 20
#define NVM_NGUID_SIZE 16
#define NVM_EUI64_SIZE 8
#define NVM_UUID_SIZE 16

struct mrvl_nvm_offload_cap {
     uint32_t nvm_version; /**< nvm version of ctrlrs exposed by DPU
*/
     uint32_t num_pcie_domains; /**< number of pcie domains */
     uint32_t num_pfs_per_domain;/**< number of pfs per domain */
     uint32_t num_vfs_per_pf; /**< number of vfs per pf */
     uint32_t total_ioq_per_pf; /**< total number of ioq pairs in a pf
*/
     uint32_t max_ioq_per_pf; /**< max number of ioq pairs allowed per
pf */
     uint32_t max_ioq_per_vf; /**< max number of ioq pairs alower per
pf */
     uint32_t max_subsystems; /**< max number of NVM subystems per DPU
*/
     uint32_t max_ns_per_subsys; /**< max namespaces per NVM subystem
*/
     uint32_t max_ctrlr_per_subsys; /**< max ctlrs per NVM subystem */
};

struct mrvl_nvm_subsys_params {
     uint8_t subnqn[NVM_SUBNQN_SIZE]; /**< subsystem nqn */
     uint8_t mn[NVM_MN_SIZE]; /**< model number */
     uint8_t sn[NVM_SN_SIZE]; /**< serial number */
     uint32_t max_namespaces; /**< maximum namespaces */
```

```
        uint16_t min_ctrlr_id;  /**< mininmum controller id */
        uint16_t max_ctrlr_id; /** maximum controller id */
};

struct mrvl_nvm_subsys_id {
        uint8_t subnqn[NVM_SUBNQN_SIZE]; /**< subsystem nqn */
};

struct mrvl_nvm_subsys_info {
        struct mrvl_nvm_subsys_params params; /**< subsystem parameters
*/
        uint32_t num_ns; /**< number of namespaces */
        uint32_t num_total_ctrlr; /**< total number of controllers */
        uint32_t num_active_ctrlr; /**< number of active controllers */
};

struct mrvl_nvm_ns_params {
        uint8_t nguid[NVM_NGUID_SIZE]; /**< nguid */
        uint8_t eui64[NVM_EUI64_SIZE]; /**< eui64 */
        uint8_t uuid[NVM_UUID_SIZE]; /**< uuid */
        uint32_t nmic; /**< nmic */
};

enum mrvl_nvm_ns_uid_type {
        NVM_NS_UID_TYPE_NGUID, /**< uid type nguid */
        NVM_NS_UID_TYPE_UUID, /**< uid type uuid */
        NVM_NS_UID_TYPE_EUI64 /**< uid type eui64 */
};

struct mrvl_nvm_ns_uid {
        enum mrvl_nvm_ns_uid_type uid_type; /**< uid type */
        union {
                uint8_t nguid[NVM_NGUID_SIZE]; /**< nguid */
                uint8_t eui64[NVM_EUI64_SIZE]; /**< eui64 */
                uint8_t uuid[NVM_UUID_SIZE]; /**< uuid */
        } u;
};

struct mrvl_nvm_ns_stats {
        uint64_t num_read_cmds; /**< num of read commands */
        uint64_t num_read_bytes; /**< num of read bytes */
        uint64_t num_write_cmds; /**< num of write commands  */
        uint64_t num_write_bytes; /**< num of write bytes */
        uint64_t num_errors; /**< number of errors */
        uint64_t total_read_latency_in_us; /**< total read latency */
        uint64_t total_write_latency_in_us; /**< total write latency
*/
        uint64_t stats_time_window_in_us;
};

struct mrvl_nvm_ns_info {
        struct mrvl_nvm_ns_params params; /**< namespace parameters */
```

```
	struct mrvl_nvm_subsys_id subsys_id; /**< subsystem id of 
namespace */
	uint32_t num_ctrlrs; /**< number of ctrlrs which have this 
namespace */
};

struct mrvl_nvm_ctrlr_stats {
	uint64_t num_admin_cmds; /**< num of admin commands recieved on 
ctrlr */
	uint64_t num_admin_cmd_errors; /**< num of admin cmd errors */
	uint64_t num_async_events; /**< num of async events sent to host 
*/
	uint64_t num_read_cmds; /**< num of read commands */
	uint64_t num_read_bytes; /**< num of read bytes */
	uint64_t num_write_cmds; /**< num of write commands */
	uint64_t num_write_bytes; /**< num of write bytes */
	uint64_t num_errors; /**< number of errors */
	uint64_t total_read_latency_in_us; /**< total read latency */
	uint64_t total_write_latency_in_us; /**< total write latency 
*/
	uint64_t stats_time_window_in_us;

};

struct mrvl_nvm_ctrlr_params {
	uint16_t pcie_domain_id, /**< pcie domain id */
	uint16_t pf_id, /**< pcie pf id */
	uint16_t vf_id, /**< pcie vf id 0 indicates PF */
	uint32_t ctrlr_id; /**< ctrlr id  */
	uint32_t max_nsq; /**< max submission queues that can be 
allocated */
	uint32_t max_ncq; /**< max completion queues that can be 
allocated */
	uint32_t mqes; /**< max num queue entries */
	uint32_t ieee_oui; /**< IEEE OUI */
	uint32_t cmic; /**< multipath capabilities */
	uint32_t nn; /**< number of namespaces */
};

struct mrvl_nvm_ctrlr_info {
	struct mrvl_nvm_ctrlr_params params;
	uint32_t active_ns_count; /**< number of active namespaces */
	uint32_t active_nsq; /**< number of submission queues */
	uint32_t active_ncq; /**< number of completion queues */
	uint32_t mdts; /**< maximum data transfer size */
	uint32_t sqes; /**< maximum sq entries */
	uint32_t cqes; /**< maximum cq entries */
};

typedef void * mrvl_dpu_handle_t;
```

```
/* DPU NODE API */

/**
 * @brief init DPU nvm functionality library
 *
 * initializes NVM functionality of DPU
 * @param handle pointer to opaque DPU Identifier object
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_nvm_init(mrvl_dpu_handle_t *handle);

/**
 * @brief get DPU nvm storage offload capabilities.
 *
 * provides capabilities of the DPU for NVMeOF offload
 * @parm handle handle to DPU node.
 * @param offload_cap pointer to set of capabilities.
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_get_offload_cap(const mrvl_nvm_handle_t handle,
                     struct mrvl_nvm_offload_cap  *offload_cap);

/**
 * @brief get NVM subystem count in DPU
 *
 * get the count of number of subystems created
 * @parm handle handle to DPU node.
 * @param subsys_count output pointer to count
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_get_subsys_count(const mrvl_nvm_handle_t handle,
                      uint32_t *subsys_count);

/**
 * @brief get list of NVM subsystems in DPU
 *
 * provides a list of created subsystems
 * @parm handle handle to DPU node.
 * @param subsys_list output pointer to list of subystems
 * @param num_input_entries number of input entries
 * @param num_output_entries actual number of output entries
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_get_subsys_list(const mrvl_nvm_handle_t handle,
                     struct mrvl_nmv_subsys_id *subsys_list,
                     uint16_t num_input_entries,
                     uint16_t *num_output_entries);
```

```
/**
 * @brief create an NVM subsystem on the DPU
 *
 * create an nvm subystem with given params.
 * User should not create subystem with duplicate SUBNQN
 * even on different DPUs
 * @parm handle handle to DPU node.
 * @param subsys_params pointer to subsystem parameters
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_create_subsystem(mrvl_nvm_handle_t handle,
                      const struct mrvl_nvm_subsys_params
*subsys_params);

/**
 * @brief remove NVM subsystem from the DPU
 *
 * remove an nvm subystem.
 * @parm handle handle to DPU node.
 * @param subsys_id pointer to subsystem id
 * All controllers and namespaces in the subsystem
 * need to removed before removing a subsystem
 * @param subsys_id pointer to subsystem_id
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_remove_subsystem(mrvl_nvm_handle_t handle,
                      const struct mrvl_nvm_subsys_id *subsys);

/**
 * @brief close DPU nvm functionality library
 *
 * close NVM functionality of DPU
 * @param handle pointer to opaque DPU Identifier object
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_nvm_close(const mrvl_dpu_handle_t *handle);

/* SUBSYSTEM API */

/**
 * @brief get subsystem info
 *
 * get subsystem information
 * @param subsys_id pointer to subsystem id
 * @param info pointer to subsystem info
 * @return 0 on sucess appropriate error code on error
 */
```

```
int
mrvl_nvm_subsys_get_info(const struct mrvl_nvm_subsys_id *subsys_id,
                 struct mrvl_nvm_subsys_info *info);

/**
 * @brief allocate namespace in a NVM subsystem
 *
 * allocates namespace in a subsystem with given params
 * @param subsys_id pointer to subsystem id
 * @param ns_params pointer to namespace parameters
 * @param bdev_name remote block device that is to be associated with
this ns
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_subsys_alloc_ns(const struct mrvl_nvm_subsys_id *subsys_id,
                 const struct mrvl_nvm_ns_params *ns_params,
                 const uint8_t *bdev_name);

/**
 * @brief unallocate namespace in a NVM subsystem
 *
 * unallocates namespace in a subsystem
 * @param subsys_id pointer to subsystem id
 * @param ns_uid pointer to ns uid
 * Namespace can be unallocated only if it is not attached to any
controller
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_subsys_unalloc_ns(const struct mrvl_nvm_subsys_id *subsys_id,
                   const struct mrvl_nvm_ns_uid *ns_uid);


/**
 * @brief get list of Namespaces in subystem
 *
 * provides a list of namespsces in subsystem
 * @param ns_list output pointer to list of namespaces
 * @param num_input_entries number of entries in input pointer
 * @param num_output_entries number of entries being returned
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_subsys_get_ns_list(const struct mrvl_nvm_subsys_id
*subsys_id,
                    struct mrvl_nvm_ns_uid *ns_list,
                    uint16_t num_input_entries,
                    uint16_t *num_output_entries);


/**
```

```
 * @brief create controller in a NVM subsystem
 *
 * creates controller in a subsystem.
 * pf_id >= 0  and vf_id = 0 indicates PF controller with pf = pf_id
 * pf_id >= 0  and vf_id >=1 indicates PF controller with pf = pf_id
and vf =
 * vf_id.
 * pf_id = -1U and vf_id = 0 indicates next available PF controller to
be used.
 * pf_id = -1U and vf_id = -1U indicates next any next available
conttroller
 * to be used (PF or VF),
 * The allocated pf_id and or vf_id will then be populated as output
parameters
 * ctrlr_id >=0  indicates ctrlr_id to be used as passed.
 * ctrlr_id = -1U indicates controller id to be allocated and filled
up
 * @param subsys_id pointer to subsystem id
 * @param ctrlr_params pointer to controller params
 * @return 0 on sucess appropriate error code on error
 */

int
mrvl_nvm_subsys_create_ctrlr(const struct mrvl_nvm_subsys_id
*subsys_id,
                     struct mrvl_nvm_ctrlr_params *ctrlr_params);

/**
 * @brief remove controller in a NVM subsystem
 *
 * removes controller in a subsystem
 * all namespaces need to be detached from the controller and
 * ctrlr should be in shutdown state before removing controller
 * @param subsys_id pointer to subsystem id
 * @param ctrlr_id controller id to be removed
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_subsys_remove_ctrlr(const struct mrvl_nvm_subsys_id
*subsys_id,
                     uint32_t ctrlr_id);

/**
 * @brief  get the list of controllers in subystem
 *
 * get the list of controllers in subystem
 * @param subsys_id pointer to subsystem id
 * @param ctrlr_id_list pointer to list of controller ids
 * @param num_input_entries number of entries in input pointer
 * @param num_output_entries number of entries being returned
 * @return 0 on sucess appropriate error code on error
 */
```

```
int
mrvl_nvm_subsys_get_ctrlr_list(const struct mrvl_nvm_subsys_id
*subsys_id,
                       int32_t *ctrlr_id_list,
                       uint32_t num_input_entries,
                       uint16_t *num_output_entries);

/* NS API */

/**
 * @brief get namespace statistics
 *
 * get the total statistics of namespace
 * @param ns_uid pointer to ns uid
 * @param stats pointer to namespace statistics
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_ns_get_stats(const struct mrvl_nvm_ns_uid *uid,
                 struct mrvl_nvm_ns_stats *stats);


/**
 * @brief  get the list of controllers attached to a namespace
 *
 * get the list of controllers attached to a namespace
 * @param ns_uid pointer to ns uid
 * @param ctrlr_id_list pointer to list of controller ids
 * @param num_input_entries number of entries in input pointer
 * @param num_output_entries number of entries being returned
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_ns_get_ctrlr_list(const struct mrvl_nvm_ns_uid *ns_uid,
                   uint32_t *ctrlr_id_list,
                   uint32_t num_input_entries,
                   uint16_t *num_output_entries);

/**
 * @brief get Namespace info
 *
 * get Namespace information
 * @param ns_uid pointer to ns uid
 * @param info pointer to ns info
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_ns_get_info(const struct mrvl_nvm_ns_uid *ns_uid,
                struct mrvl_nvm_ns_info *info);

/* CTRLR API */
```

```
/**
 * @brief attach namespace to a controller in  NVM subsystem
 *
 * attach namespace to a controller
 * if this is not the first attachment in the subystem
 * nsid should match the nsid used for first attachment
 * this generates an async event to the host with namespace attribute
changed
 * notification causing hotplug of namespace.
 * @param subsys_id pointer to subsystem id
 * @param ns_uid pointer to ns uid
 * @param nsid nsid to be used for this namespace in this controller
 * @return 0 on sucess appropriate error code on error
 */

mrvl_nvm_ctrlr_attach_ns(const struct mrvl_nvm_subsys_id *subsys_id,
                 uint32_t ctrlr_id,
                 const struct mrvl_nvm_ns_uid *ns_uid, uint32_t nsid,
                 uint8_t nmic);

/**
 * @brief detach namespace from a controller in  NVM subsystem
 *
 * detach namespace to a controller
 * @param subsys_id pointer to subsystem id
 * @param nsid nsid of the  namespace
 * this generates an async event to the host with namespace attribute
changed
 * notification causing hotunplug of namespace.
 * ctrlr should be in shutdown state before removing controller
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_ctrlr_detach_ns(const struct mrvl_nvm_subsys_id *subsys_id,
                 uint32_t ctrlr_id, const struct mrvl_nvm_ns_uid
*uid);
/**
 * @brief  get controller info
 *
 * get the controller info
 * @param subsys_id pointer to subsystem id
 * @param ctrlr_id controller id
 * @param info pointer to controller info
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_ctrlr_get_info(const struct mrvl_nvm_subsys_id *subsys_id,
                uint32_t ctrlr_id,
                struct mrvl_nvm_ctrlr_info *info);
/**
 * @brief  get controller statistics
 *
 * get the statistics of a controller
 * @param subsys_id pointer to subsystem id
 * @param ctrlr_id controller id
 * @param stats pointer to controller stats
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_ctrlr_get_stats(const struct mrvl_nvm_subsys_id *subsys_id,
                 uint32_t ctrlr_id,
                 struct mrvl_nvm_ctrlr_stats *stats);

/**
 * @brief  get ctrlr namespace statistics
 *
 * get the statistics of namespace for this controller
 * Note, these are per this controller only, do not account for the
 * namespace usage in other controllers.
 * @param subsys_id pointer to subsystem id
 * @param ctrlr_id controller id
 * @param stats pointer to namespace statistics
 * @return 0 on sucess appropriate error code on error
 */
int
mrvl_nvm_ctrlr_get_ns_stats(const struct mrvl_nvm_subsys_id
*subsys_id,
                    uint32_t ctrlr_id,
                    const struct mrvl_nvm_ns_uid *uid,
                    struct mrvl_nvm_ns_stats *stats);

#endif /* __MRVL_NVM_API_H__ */
```

Code	**Description**
Codes already supported in spdk json-rpc	
-1	Invalid state - given method exists but it is not callable in current runtime state
-32600	Invalid request - not compliant with JSON-RPC 2.0 Specification
-32601	Method not found
-32602	Invalid params
-32603	Internal error for e.g.: errors like out of memory
-32700	Parser error
Codes supported by Mrvl SPDK Application json-rpc	
-32500	Invalid namespace object id
-32501	Namespace cannot be freed as it is attached to a controller.

Error-Codes Table 1