NVMe™ and PCIe SSDs
NVMe™ Management Interface

Peter Onufryk
Sr. Director, Product Development
PMC-Sierra

Austin Bolen
Storage Development Principal Engineer
Dell

Special thanks to the NVMe™ Management Interface Workgroup members for contributions & support.
Agenda

- NVMe Management Ecosystem
- In-band vs Out-of-Band Management
- NVMe Out-of-Band Management Stack Overview
  - Transport Layer (MCTP)
  - Protocol Layer (NVMe Management Command Set)
- NVMe Device
  - Management Architectural Model
  - Command Processing
- Mgmt. Controller/Host Communication
- Summary
NVMe Storage Device Management

Server Caching
- Root Complex
- PCIe Switch
- NVMe
- x16
- NVMe
- x4
- NVMe
- NVMe

Server Storage
- Root Complex
- PCIe/PCIe RAID
- NVMe
- x16
- NVMe
- x4
- NVMe
- NVMe
- NVMe

External Storage
- Controller A
- SAS
- PCIe Switch
- Root Complex
- x16
- Controller B
- SAS
- PCIe Switch
- Root Complex
- x16

- Example Pre-boot Management
  - Inventory, Power Budgeting, Configuration, Firmware Update
- Example Out-of-Band Management During System Operation
  - Health Monitoring, Power/Thermal Management, Firmware Update, Configuration
**NVMe Architecture (review)**

- **NVM Subsystem** - one or more controllers, one or more namespaces, one or more PCI Express ports, a non-volatile memory storage medium, and an interface between the controller(s) and non-volatile memory storage medium.

*NVM Subsystem with One Controller and One Port*

*NVM Subsystem with Two Controllers and Two Ports*
An NVMe Storage Device consists of one NVM Subsystem with:
- One or more PCIe ports
- An optional SMBus/I2C interface
Driver vs. Out-of-Band Management

- Applications
- Operating System
- Management Controller
- Platform Management

Components:
- NVMe Driver
- NVMe Driver
- PCIe
- PCIe VDM
- PCIe VDM
- SMBus/I2C
- PCIe Port 0
- PCIe Port 1
- NVM Subsystem
- PCIe SSD
Management Interface Protocol Layering

- **Management Applications (e.g., Remote Console)**
- **Management Controller (BMC or Host Processor)**
- **NVMe Management Interface**
- **Management Component Transport Protocol (MCTP)**
  - MCTP over SMBus/I2C Binding
  - MCTP over PCIe VDM Binding
- **SMBus/I2C**
- **PCIe**
- **PCle SSD**
MCTP Terminology

- MCTP defines a communication model used to transfer data between management entities.

- Management Controller (MC): A microcontroller or processor that aggregates management parameters from one or more management devices and makes access to those parameters available to local or remote software.

- Management Device: A device managed by a Management Controller.

- MCTP Packet: Base unit of transfer in MCTP.

- MCTP Message: One or more MCTP Packets.
MCTP Packet - SMBus/I²C

- **Byte 1**: Destination Slave Address
- **Byte 2**: Command Code (MCTP = 0Fh), Byte Count, Source Slave Address
- **Byte 3**: Destination Endpoint ID, Source Endpoint ID
- **Byte 4**: MCTP Reserved,Hdr Version, Destination Endpoint ID
- **Byte 5**: Source Endpoint ID
- **Byte 6**: Message Type
- **Byte N**: PEC

---

**MCTP Message Header (Varies based on Message Type)**

**Message Type byte**
(only required in first packet header of message.)

= common fields for all MCTP messages
MCTP Packet - PCIe VDM

Pad Length - (2-bits) indicates # of pad bytes to get PCIe message DWORD aligned.

MCTP defined usage of PCIe TAG field

PCI Requester ID

PCI Target ID (for Route by ID messages, Otherwise reserved)

Vendor ID - DMTF

MCTP Message Header (Varies based on Message Type, Message Type byte only present in first packet header of message.)

MCTP Message Data (Can span multiple packets.)

Message Integrity Check

PCIe Medium - specific Trailer

MCTP VDM Code uniquely identifies MCTP VDMs from other possible VDMs that may be defined under the DMTF Vendor ID
# NVMe Management Interface Command Set Overview (preliminary)

<table>
<thead>
<tr>
<th>Command Type</th>
<th>Command</th>
<th>Command Type</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVMe Management Interface Specific Commands</td>
<td>Controller Inventory</td>
<td>NVMe Commands</td>
<td>Get Log Page</td>
</tr>
<tr>
<td></td>
<td>Read / Write VPD</td>
<td></td>
<td>Identify</td>
</tr>
<tr>
<td></td>
<td>Run Diagnostics</td>
<td></td>
<td>Set Feature</td>
</tr>
<tr>
<td></td>
<td>Health Status</td>
<td></td>
<td>Get Feature</td>
</tr>
<tr>
<td></td>
<td>Command Flow Control</td>
<td></td>
<td>Firmware Activate</td>
</tr>
<tr>
<td></td>
<td>Exception Handling</td>
<td></td>
<td>Firmware Image Download</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Vendor Specific</td>
</tr>
<tr>
<td>PCIe Command</td>
<td>Configuration Read</td>
<td></td>
<td>Format NVM</td>
</tr>
<tr>
<td></td>
<td>Configuration write</td>
<td></td>
<td>Security Send</td>
</tr>
<tr>
<td></td>
<td>I/O Read</td>
<td></td>
<td>Security Receive</td>
</tr>
<tr>
<td></td>
<td>I/O Write</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Memory Read</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Memory Write</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## NVMe Management Interface Specific Commands

- Controller Inventory
- Read / Write VPD
- Run Diagnostics
- Health Status
- Command Flow Control
- Exception Handling

## PCIe Command

- Configuration Read
- Configuration write
- I/O Read
- I/O Write
- Memory Read
- Memory Write

## NVMe Commands

- Get Log Page
- Identify
- Set Feature
- Get Feature
- Firmware Activate
- Firmware Image Download
- Vendor Specific
- Format NVM
- Security Send
- Security Receive

## Exception Handling

...
**NVM Subsystem Architectural Model**

- **NVMe Management Endpoint** – An MCTP endpoint that is the terminus and origin of MCTP packets/messages and which processes MCTP and NVMe Management Interface commands

- **NVMe Controller Management Interface** – An interface associated with each NVMe controller in the NVM subsystem that is responsible for executing management operation on behalf on an NVMe Management Endpoint
1. Management Controller sends request message to NVMe Management Endpoint
2. Management Endpoint processes command
3. NVMe Management Endpoint sends response message to Management Controller
1. Management Controller sends request message to NVMe Management Endpoint
2. NVMe Management Endpoint forwards request to appropriate NVMe Controller Management Interface
3. NVMe Controller Management Interface executes command on NVMe Controller
4. NVMe Management Endpoint sends response back to NVMe Management Endpoint
5. NVMe Management Endpoint sends response message to Management Controller
Mgmt. Controller (MC)/Host Communication

- MC needs data/notification from host OS, driver, or app
  - Driver version
  - Software/OS RAID information
  - OS name of device (e.g. “/dev/nvme0n1” or “\\PhysicalDrive1”)
  - Host changes device configuration (UI)

- MC needs to send data/notification to host OS, driver, or app (UI)
  - MC initiated f/w update is in progress
  - Prepare device for hot removal

- Synchronize access to a shared resource on the NVMe device (UI)
  - Changing power states
  - Setting thresholds (temperature, spare blocks)
Sending Data from Host to MC

- Use existing NVMe Set/Get Features commands
- New Feature Identifiers reserved in NVMe spec.
- Format of each Feature Identifier defined in NVMe Management Interface spec.

Management Feature Identifiers:
- NVMe Controller Metadata
- NVMe Namespace Metadata
## Host Data Format

### Type-Length-Value (TLV) Element Structure

<table>
<thead>
<tr>
<th>Type + Version (2 bytes)</th>
<th>Length (2 Bytes)</th>
<th>Value (Length Bytes)</th>
</tr>
</thead>
</table>

- **TLV elements**
  - Stored as a list in Get/Set Features Data Structure Element
  - First element at offset 0, second element at offset 4 + Length of first element, etc.

- A value of ‘0’ for the Type is used as a terminator value to the end the TLV element list
# Controller Metadata

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0h</td>
<td>End of TLV Elements</td>
</tr>
<tr>
<td>1h</td>
<td>Feature ID Specific Data</td>
</tr>
<tr>
<td>2h</td>
<td>Operating System Controller Name</td>
</tr>
<tr>
<td>3h</td>
<td>OS Driver Name (ODN)</td>
</tr>
<tr>
<td>4h</td>
<td>OS Driver Version (ODV)</td>
</tr>
<tr>
<td>5h</td>
<td>Pre-boot Driver Name (PDN)</td>
</tr>
<tr>
<td>6h</td>
<td>Pre-boot Driver Version (PDV)</td>
</tr>
<tr>
<td>7h</td>
<td>Current State (Offline, Online, Prepared for Removal, etc.)</td>
</tr>
</tbody>
</table>

---

Controller Metadata

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0h</td>
<td>End of TLV Elements</td>
</tr>
<tr>
<td>1h</td>
<td>Feature ID Specific Data</td>
</tr>
<tr>
<td>2h</td>
<td>Operating System Controller Name</td>
</tr>
<tr>
<td>3h</td>
<td>OS Driver Name (ODN)</td>
</tr>
<tr>
<td>4h</td>
<td>OS Driver Version (ODV)</td>
</tr>
<tr>
<td>5h</td>
<td>Pre-boot Driver Name (PDN)</td>
</tr>
<tr>
<td>6h</td>
<td>Pre-boot Driver Version (PDV)</td>
</tr>
<tr>
<td>7h</td>
<td>Current State (Offline, Online, Prepared for Removal, etc.)</td>
</tr>
</tbody>
</table>
## Namespace Metadata

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0h</td>
<td>End of TLV Elements</td>
</tr>
<tr>
<td>1h</td>
<td>Feature Identifier Specific Data</td>
</tr>
<tr>
<td>2h</td>
<td>Operating System Namespace Name</td>
</tr>
<tr>
<td>3h</td>
<td>RAID Information</td>
</tr>
<tr>
<td>4h</td>
<td>Caching Information</td>
</tr>
</tbody>
</table>
## Sample Controller Metadata

<table>
<thead>
<tr>
<th>Offset</th>
<th>Contents</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0      | \([15:12] = 0\) \([11:00] = 5\) | TLV Element 1 Revision  
               TLV Element 1 Type (Preboot Driver Name) |
| 2      | 16                   | TLV Element 1 Length                                                        |
| 4      | UEFI NVMe Driver     | TLV Element 1 Value                                                         |
| 20     | \([15:12] = 0\) \([11:00] = 6\) | TLV Element 2 Revision  
               TLV Element 2 Type (Preboot Driver Version) |
| 22     | 7                    | TLV Element 2 Length                                                        |
| 24     | 1.2.3.4              | TLV Element 2 Value                                                         |
| 31     | \([15:12] = 0\) \([11:00] = 0\) | TLV Element 3 Revision  
               TLV Element 3 Type is 0. End of list. |
Summary

- We are standardizing out-of-band management interface for NVMe storage devices
  - PCIe VDM and SMBus/I2C

- The NVMe management interface is leveraging other management specifications/standards
  - Complementary and not a replacement

- The specification is planned to be completed at the end of this year
References

- MCTP Overview:

- MCTP Base Spec:
  http://www.dmtf.org/sites/default/files/standards/documents/DSP0236_1.2.0.pdf

- MCTP SMBus/I2C Binding:
  http://www.dmtf.org/sites/default/files/standards/documents/DSP0237_1.0.0.pdf

- MCTP over PCIe VDM Overview:

- MCTP PCIe VDM Binding:
  http://www.dmtf.org/sites/default/files/standards/documents/DSP0238_1.0.1.pdf

- IPMI Platform Management FRU Information Storage Definition:
Backup
Vital Product Data (VPD)

- Vital Product Data (VPD) contains information about the storage device
  - Examples:
    - Manufacturer
    - Serial number
    - Device configuration
    - Power requirements
  - See IPMI FRU information
- VPD accessible using I2C serial EEPROM read/write operations and NVMe Management Interface commands over MCTP
- Two I2C addresses
  - I2C serial EEPROM access (VPD device)
  - MCTP Endpoint (NVMe controller ASIC)
- VPD accessibility during power modes
  - During Auxiliary Power
    - I2C serial EEPROM read/write
  - During Main Power
    - I2C serial EEPROM read/write
    - NVMe Management Interface commands
SMBus/I2C Topologies

Shared SMBus/I2C

- Management Controller
- Other SMBus Device
- Other SMBus Device
- NVMe SSD
- NVMe SSD
- NVMe SSD

Requires Unique SMBus/I2C addresses

Segmented SMBus/I2C

- Management Controller
- Mux Control
- SMBus/I2C Mux
- Other SMBus Device
- Other SMBus Device
- NVMe SSD
- NVMe SSD
- NVMe SSD
- NVMe SSD

Repeated SMBus/I2C Addresses Supported
SMBus/I2C Addressing

- **During Auxiliary Power (if supported)**
  - I2C serial EEPROM read/write access at default SMBus/I2C address 0xA6, but may be modified using ARP

- **During Main Power**
  - MCTP Endpoint at default SMBus/I2C address 0xD4, but may be modified using ARP
  - I2C serial EEPROM read/write access
    - If auxiliary power was provided, then SMBus/I2C address shall be maintained if modified using ARP; otherwise, the default address is 0xA6
    - SMBus/I2C address may be modified using ARP

- Supports both shared and segmented SMBus/I2C environments