LEGAL NOTICE:

This NVM Express Management Interface revision 1.0a technical proposal is proprietary to the NVM Express, Inc. (also referred to as "Company") and/or its successors and assigns.

NOTICE TO USERS WHO ARE NVM EXPRESS, INC. MEMBERS: Members of NVM Express, Inc. have the right to use and implement this NVM Express Management Interface revision 1.0a technical proposal subject, however, to the Member's continued compliance with the Company's Intellectual Property Policy and Bylaws and the Member's Participation Agreement.

NOTICE TO NON-MEMBERS OF NVM EXPRESS, INC.: If you are not a Member of NVM Express, Inc. and you have obtained a copy of this document, you only have a right to review this document or make reference to or cite this document. Any such references or citations to this document must acknowledge NVM Express, Inc. copyright ownership of this document. The proper copyright citation or reference is as follows: **"© 2007 - 2019 NVM Express, Inc. ALL RIGHTS RESERVED."** When making any such citations or references to this document you are not permitted to revise, alter, modify, make any derivatives of, or otherwise amend the referenced portion of this document in any way without the prior express written permission of NVM Express, Inc. Nothing contained in this document shall be deemed as granting you any kind of license to implement or use this document or the specification described therein, or any of its contents, either expressly or impliedly, or to any intellectual property owned or controlled by NVM Express, Inc., including, without limitation, any trademarks of NVM Express, Inc.

LEGAL DISCLAIMER:

THIS DOCUMENT AND THE INFORMATION CONTAINED HEREIN IS PROVIDED ON AN "**AS IS**" BASIS. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, NVM EXPRESS, INC. (ALONG WITH THE CONTRIBUTORS TO THIS DOCUMENT) HEREBY DISCLAIM ALL REPRESENTATIONS, WARRANTIES AND/OR COVENANTS, EITHER EXPRESS OR IMPLIED, STATUTORY OR AT COMMON LAW, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, VALIDITY, AND/OR NONINFRINGEMENT.

All product names, trademarks, registered trademarks, and/or service marks may be claimed as the property of their respective owners.

NVM Express Management Interface Workgroup
c/o VTM Inc.
3855 SW 153rd Drive
Beaverton, OR 97003
info@nvmexpress.org

NVM Express Technical Proposal for New Feature

Technical Proposal ID	**6003 – Multi NVM Subsystem Management**
Change Date	**3/20/2019**
Builds on Specification	**NVM Express Management Interface 1.0a**

Technical Proposal Author(s)

Name	Company
Myron Loewen	Intel Corporation
Peter Onufryk	Microsemi

The purpose of this Technical Proposal is to specify the mechanisms and architecture necessary to discover the presence of NVMe Storage Devices that contain multiple NVM Subsystems and/or other elements (e.g., PCIe switches and expansion connectors) and enable management of these NVMe Storage Devices.

The mechanism supports both muxed and shared SMBus/I2C topologies with either default addressing or ARP. VPD enhancements are defined that allow discovery of the number of NVM Subsystems contained in an NVMe Storage Device and their connections. A new SMBus/I2C address is also reserved for FRU Information Devices to prevent conflicts when NVMe Storage Devices are plugged into expansion connectors.

For NVMe Storage Devices with multiple NVM Subsystems that support an SMBus/I2C port, NVMe-MI traffic is directed to a specific NVM Subsystem using either an SMBus/I2C Mux or a unique SMBus address assigned via ARP.

Revision History

Revision Date	Change Description
10/23/2017	First published draft based on prior power point discussions
01/15/2018	All technical feedback incorporated and ready for wordsmithing
02/28/2018	Split parameterized link records into separate data types with new label record and simplified SMBus/I2C Mux sorting constraints. Updated Figure 109 for Vaux off state.
04/16/2018	Cleaned up syntax, vendor defined link types, PCIe reserved link types, and SMBus devices table
05/28/2018	Reversed several ideas based on last NVMe-MI meeting. Keeping old NVMe Ports MultiRecord and adding one for Elements. Clarified A6h VPD address to be for all leaf node devices, ie SSDs that do not have Expansion Connectors.
07/15/2018	Shuffled the bytes around in the new PCIe Elements MultiRecord for clearer descriptions of devices and somewhat cleaner backwards compatibility. This eliminated the labels MultiRecord and separated the prior MultiRecords for NVMe-MI 1.0 devices and the new MultiRecord for devices with multiple NVM subsystems or expansion ports.
8/26/2018	Changed the disconnected link indicator from FFh to 00h. Rewrote the PCIe Retimer Descriptor to match real devices. Many syntax improvements.
9/5/2018	Combined the two Expansion Connector types, eliminated element ordering requirements, added definition for expansion power connector mapping and integrated all known syntax feedback
9/28/2018	Changed PCIe Elements MultiRecord to the more descriptive name of Topology MultiRecord. Rewrote all Topology Descriptors Figures. Rewrote descriptive text in sections 9.2.5 through 9.2.5.2. Relaxed strict size requirements in Product Info Area and eliminated NVM Capacity. Highlighted all references in yellow that need hyperlinks. Moved Extended Element Descriptor to Type 1. Deleted Element Descriptors for SMBus/I2C Wire Bridge and PCIe Retimer.
11/14/2018	Eliminated Appendix D, printf formatting of label strings, and Element descriptor for power. Many syntax improvements.
11/17/2018	Many additional syntax improvements, changed Label Element encoding from ASCII to UTF-8, gave ARP and Mux solutions equal recommendation, and renamed Host Connector to Upstream Connector.
11/21/2018	Lots of syntax improvements from multiple reviewers. Changed Basic ARP to make it more specific for multi NVM subsystems. Made VPD requirements more explicit for permanently populated Expansion Connectors.
12/14/2018	Removed technical author at author's request.
3/20/2019	Ratified

Description of Specification Changes

Editor's note: These changes interact with TP6001 & TP6002 and an attempt was made to use their merged content as a base for this document, the section numbers correspond to merged content in the current NVMe-MI 1.1 draft. Note that yellow highlight is used to indicate all references that need hyperlinks. Figures and Tables with TBD values should be replaced with the correct sequential reference number in the final document.

Modify Section 1.2.1 as shown below:

The management of NVMe over Fabrics ~~Storage Devices or NVMe Enclosures containing multiple architecturally visible NVM Subsystems~~ is outside the scope of this specification. This specification does not define new security mechanisms.

Modify Section 1.4.1 as shown below and add 2 figures:

An ~~NVMe Storage Device Field Replaceable Unit, or simply~~ NVMe Storage Device that implements the out-of-band mechanisms but not the in-band mechanisms defined in this specification consists of ~~an~~zero or more NVM Subsystems. An NVMe Storage Device that implements the in-band mechanisms defined in this specification consists of one or more NVM Subsystems. ~~that~~ Each NVM Subsystem includes ~~a non-volatile storage medium along with~~ one or more Management Endpoints. ~~T~~In these NVM Subsystems there may be up to one Management Endpoint per PCIe port and up to one Management Endpoint per SMBus/I2C port. Each Management Endpoint has a Port Identifier that is less than or equal to the Number of Ports (NUMP) field value in the NVM Subsystem Information Data Structure.

An NVMe Storage Device that is a Field-Replaceable Unit (FRU) is a physical component, device, or assembly that a technician can remove and replace without having to replace the entire system in which it is contained. Examples of NVMe Storage Device Field-Replaceable Units include a U.2 PCIe SSD, a PCI Express Card Electromechanical (CEM) add-in card, and an M.2 module. The FRU referenced by the FRU Globally Unique Identifier (FGUID) field in the NVM Express Specification shall be an NVMe Storage Device Field-Replaceable Unit.

There are many variants of an NVMe Storage Device. One example is an NVMe Storage Device that only contains a single NVM Subsystem. Another example may contain no NVM Subsystems and instead have one or more Expansion Connectors for adding additional NVMe Storage Device FRUs. Such an NVMe Storage Device is referred to as a Carrier. In another example, the NVMe Storage Device may contain one or more NVM Subsystems and one or more Expansion Connectors. NVMe Storage Devices may contain PCIe switches which connect to one or more NVM Subsystems or Expansion Connectors. NVMe Storage Devices may contain SMBus/I2C Muxes that connect to one or more NVM Subsystems or Expansion Connectors.

NVMe-MI supports Vital Product Data (VPD) that utilizes the format defined in the IPMI Platform Management FRU Information Storage Definition and is stored in a FRU Information Device. The FRU Information Device may be implemented in the NVM Subsystem, in an external device (e.g., serial EEPROM), or a combination of the two. The VPD is accessible over any port that supports NVMe-MI using MCTP commands. If the NVMe Storage Device has an SMBus/I2C port~~interface~~, then the VPD is accessible using the access mechanism over I2C as defined in the IPMI Platform Management FRU Information Storage Definition.

An NVMe Storage Device is required to have a FRU Information Device for each Upstream Connector that supports SMBus/I2C. If an NVMe Storage Device contains multiple NVM Subsystems, then the FRU Information Device associated with each NVM Subsystem is optional since the required FRU Information Device connected to the Upstream Connector describes the entire NVMe Storage Device. The contents of these optional FRU Information Devices is out of scope for this specification.

Figure 1 illustrates an NVMe Storage Device that is a single-port PCIe SSD with the FRU Information Device at address A6h implemented by the NVM Subsystem. Figure illustrates an NVMe Storage Device that is a dual-port PCIe SSD with an SMBus/I2C port and a FRU Information Device at address A6h implemented using a Serial EEPROM.

Figure 1: Single-Port PCIe SSD

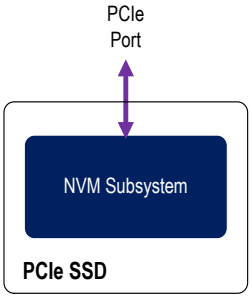
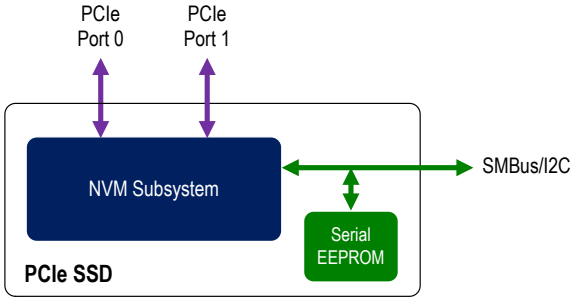


Figure 2: Dual-Port PCIe SSD with SMBus/I2C



An example U.2 form factor NVMe Storage Device with Expansion Connectors (i.e. a Carrier) is shown in Figure TBD1. This Carrier has two M.2 Expansion Connectors for connecting two M.2 NVMe Storage Device FRUs. The Carrier and each M.2 NVMe Storage Device are separate NVMe Storage Device FRUs, each with their own FRU Information Device. The FRU Information Device on the Carrier is at address A4h and the FRU Information Devices on each M.2 NVMe Storage Device has a default address of A6h and supports the SMBus Address Resolution Protocol (ARP). ARP is used after power is applied to reassign the conflicting A6h addresses before the M.2 FRU Information devices are read. ARP would also be used to reassign the conflicting MCTP addresses and potentially additional elements.

Figure TBD1: NVMe Storage Device with Connectors (i.e., a Carrier)

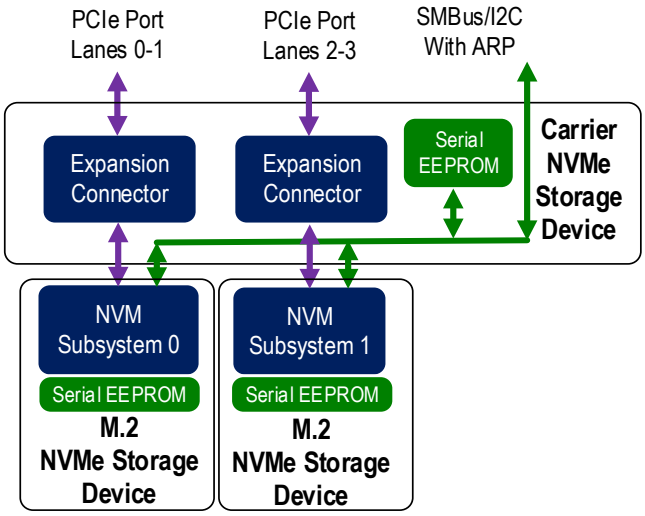
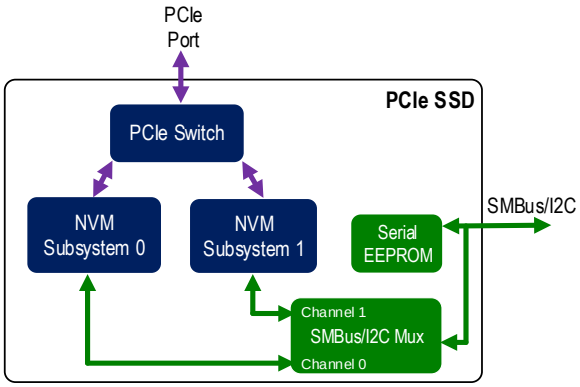


Figure TBD2 shows an NVMe Storage Device that contains two NVM Subsystems implemented using soldered down BGA packages and a FRU Information Device at address A6h implemented using a Serial EEPROM. An NVMe Storage Device without Expansion Connectors that implements an SMBus/I2C port always contains a FRU Information Device at address A6h directly connected to the Upstream Connector. An SMBus/I2C Mux is used in this example instead of ARP to eliminate SMBus/I2C address collisions. The SMBus/I2C Mux is configured by a Management Controller prior to communications with the desired NVM Subsystem. The FRU Information Device contains the details necessary to configure the SMBus/I2C Mux.

Figure TBD2: NVMe Storage Device with two NVM Subsystems and an SMBus/I2C Mux



The NVMe Management Interface is used to send Command Messages which consist of standard NVMe Admin Commands that target a Controller within the NVM Subsystem; commands that provide access to the PCI Express configuration ….

Modify Section 1.7 as shown below (Note: order the items in alphabetical order):

1.7.x Carrier

An NVMe Storage Device FRU with one or more Expansion Connectors and zero or more NVM Subsystems.

1.7.x Expansion Connector

A connector where an NVMe Storage Device FRU or cable may be attached to a Carrier. Expansion connectors may be empty or populated.

1.7.x Field Replaceable Unit (FRU)

A physical component, device, or assembly in a system that an end user or technician can remove and replace without having to replace the entire system in which it is contained. The Field-Replaceable Unit described in this specification is an NVMe Storage Device Field-Replaceable Unit (refer to 1.7.x).

1.7.13 NVMe Enclosure (Enclosure)

A platform, card, module, box, rack, or set of boxes that may provide power, cooling, mechanical protection and/or external interfaces for zero or more NVMe Storage Devices. An Enclosure may itself contain one or more NVM Subsystems and shall contain one or more Enclosure Services Processes.

1.7.x NVMe Storage Device

A logical or physical component, device, or assembly that contains at least one NVM subsystem or Expansion Connector and at least one Upstream Connector. It may contain additional elements such as: FRU Information Devices, PCIe switches, and SMBus/I2C Muxes. An NVMe Storage Device shall comply with the NVM Express specification. In this specification, NVMe Storage Devices shall also comply with this specification.

1.7.x NVMe Storage Device FRU

An NVMe Storage Device that an end user or technician can remove and replace without having to replace the entire system in which it is contained. Examples of NVMe Storage Device Field-Replaceable Units include a U.2 PCIe SSD, a PCI Express Card Electromechanical add-in card, or an M.2 module.

1.7.x SMBus/I2C Mux

A bidirectional SMBus/I2C fan-out multiplexer with one upstream channel and one or more downstream channels configured by an I2C command from a Management Controller to connect specific channels. Each channel may be connected to devices with SMBus/I2C ports. This multiplexer permits multiple devices to use the same SMBus/I2C addresses as long as they are on separate channels.

1.7.x Upstream Connector

A connector on the NVMe Storage Device to which a host or Management Controller attaches. It may be a physical connector as in U.2 form factors, solder balls as in a BGA form factor, or PCB trace fingers as in a CEM Add in Card or EDSFF form factor. An Upstream Connector may include multiple communications ports, control signals, and power supply rails.

Modify Section 2.2 as shown below:

2.2 SMBus/I2C

This section defines the requirements for an NVMe Storage Device that implements an SMBus/I2C Port. The SMBus/I2C physical layer is only applicable for the out-of-band mechanism.

The NVMe Storage Device shall contain a FRU Information Device as defined in the IPMI Platform Management FRU Information Storage Definition specification associated with each Upstream Connector.

If an NVM Subsystem implements a Management Endpoint associated with the SMBus/I2C port ~~If the NVM Subsystem implements an SMBus/I2C interface and associated with that SMBus/I2C interface is a Management Endpoint,~~ then that ~~interface~~port shall comply to ~~MCTP over SMBus/I2C as specified by~~ the Management Component Transport Protocol (MCTP) SMBus/I2C Transport Binding Specification.

~~The SMBus/I2C physical layer is only applicable in the out-of-band mechanism. If anthe NVMe Storage Device implements an SMBus/I2C interface, then the~~An NVM Subsystem may optionally support the NVMe Basic Management Command for health and status polling. The NVMe Basic Management Command is defined in Appendix A and is not recommended for new designs. It is possible to support both MCTP and the Basic Management Command.

Figure TBD3 lists SMBus/I2C elements that are supported on an NVMe Storage Device. For each SMBus/I2C element, the default SMBus/I2C address is provided as well as the conditions under which the SMBus/I2C element is required on an NVMe Storage Device. The presence or absence of Expansion Connectors on an NVMe Storage Device determines which of the two mutually exclusive SMBus/I2C addresses is used for the FRU Information Device. Using different SMBus/I2C address for the FRU Information Device on NVMe Storage Devices that are Carriers versus non-Carriers avoids SMBus/I2C address conflicts when Expansion Connectors are populated with NVMe Storage Devices.

ARP support on SMBus/I2C elements is optional but is required to avoid SMBus/I2C address conflicts if multiple SMBus/I2C elements with the same default SMBus/I2C address are present on the same SMBus/I2C channel.

Figure TBD3: SMBus/I2C Elements and Requirements

SMBus/I2C Element	Default SMBus/I2C Address		SMBus ARP Support	Required Element Presence
	Hex Format	Binary Format[1]		
FRU Information Device	A6h	1010_011xb	Optional	Required on an NVMe Storage Device with **no** Expansion Connectors.
FRU Information Device	A4h	1010_010xb	Optional	Required on Carriers (i.e. an NVMe Storage Device with **one or more** Expansion Connectors).
NVM subsystem SMBus/I2C Management Endpoint	3Ah	0011_101xb	Optional	Required if an NVM Subsystem has an SMBus/I2C Management Endpoint.
SMBus/I2C Mux	E8h	1110_100xb	Optional	Required if there is more than one SMBus/I2C element on any SMBus/I2C channel with the same SMBus/I2C address that does not support ARP.
NVM subsystem Basic Management Command (Appendix A)	D4h	1101_010xb	Optional	Not recommended for new designs.

Notes:
1. The x represents the SMBus/I2C read/write bit.

. ~~The address 3Ah appears on SMBus as 0011_101xb where x represents the SMBus read/write bit.~~

Host platforms expecting to be used with one or more Management Endpoints (e.g., data center platforms and workstations) should isolate SMBus/I2C ~~segments~~channels to avoid a Management Endpoint conflicting with the address of another SMBus/I2C element. An SMBus/I2C address conflict may occur when a Management Endpoint that does not support ARP is used with platforms that do not isolate SMBus/I2C channels~~segments~~ (e.g., some client platforms). ARP may be used to dynamically reassign SMBus/I2C addresses in a system when supported by both the Management Controller and the NVMe Storage Devices.

SMBus/I2C elements on an NVMe Storage Device that support ARP should be implemented as Default Slave Address (DSA) devices as defined by the SMBus specification. These devices should not issue "Notify ARP Master" commands because the NVMe Storage Device should not initiate SMBus/I2C traffic.

~~The SMBus/I2C Management Endpoint shall be accessible at a power-up SMBus/I2C address of 3Ah and should be SMBus ARP-capable (as defined in the SMBus 3.0 specification).[1] If the NVM Subsystem is "Discoverable" (as defined in the SMBus 3.0 specification), the device may issue a "Notify ARP Master" command when the NVM Subsystem is ready to communicate. If the NVM Subsystem implements an SMBus/I2C interface, then VPD information shall be accessible from the Management Endpoint using Sequential Read and Random Read operations as defined by the IPMI Platform Management FRU Information Storage Definition specification.~~

~~The VPD shall be accessible using I2C read operations from a FRU Information Device at a power-up SMBus/I2C address of A6h and should be SMBus ARP-capable (as defined in the SMBus 3.0 specification).[2] If the FRU Information Device is "Discoverable" (as defined in the SMBus 3.0 specification), it may issue a "Notify ARP Master" command when the FRU Information Device is ready to communicate.~~

If ARP is supported by an NVM Subsystem, then all SMBus/I2C elements~~the NVM Subsystem and the FRU Information Device~~ associated with that NVM Subsystem~~, if present, SMBus/I2C Management Endpoint and VPD~~ shall ~~both~~ use the SMBus Address Resolution Protocol Unique Device Identifier (UDID) shown in Modify Figure ***7 to define additional UDID use cases shown below:***

Figure 3. The ARP UDID is a unique identifier created by the NVMe Storage Device vendor. The UDID Vendor ID bits 30 and 31 allow up to four SMBus/I2C elements to be grouped together with the same NVM Subsystem. The only difference within this group of ~~between~~ UDIDs~~the NVM Subsystem and FRU Information Device UDID~~ is the most significant two bits of the Vendor Specific ID. This fact may be used by the Management Controller~~MCTP bus owner~~ to associate an SMBus/I2C Management Endpoint with its corresponding VPD.

If there are multiple NVM Subsystems in an SMBus ARP-capable NVMe Storage Device, then the Unique NVM Storage Device ID field of the UDID shall be incremented by one for each NVM Subsystem. If the Upstream Connector has an SMBus/I2C port, then the FRU Information Device associated with that connector shall be present directly on the SMBus/I2C channel connected to the Upstream Connector.

Modify Figure 7 to define additional UDID use cases shown below:

~~[1] The address 3Ah appears on SMBus as 0011_101xb where x represents the SMBus read/write bit.~~
~~[2] The address A6h appears on SMBus as 1010_011xb where x represents the SMBus read/write bit.~~

Figure 3: ~~NVM Subsystem and FRU Information Device SMBus~~ SMBus/I2C Element UDID

<table>
<tr><th>Bits</th><th>Field</th><th>Description</th></tr>
<tr><td>127:120</td><td>Device Capabilities</td><td>This field describes the device capabilities

<table><tr><th>Bits</th><th>Description</th></tr><tr><td>7:6</td><td>Address Type: This field describes the type of address contained in the device. Refer to the SMBus transport binding specification.</td></tr><tr><td>5:1</td><td>Reserved</td></tr><tr><td>0</td><td>PEC Supported: All MCTP transactions shall include a Packet Error Code (PEC) byte. This field shall be set to one to indicate support for PEC.</td></tr></table></td></tr>
<tr><td>119:112</td><td>Version / Revision</td><td>This field is used to identify the UDID version and silicon revision.

<table><tr><th>Bits</th><th>Description</th></tr><tr><td>7:6</td><td>Reserved</td></tr><tr><td>5:3</td><td>UDID Version. This field specifies the UDID version and shall be set to 001b</td></tr><tr><td>2:0</td><td>Silicon Revision ID: This field is used to specify a vendor specific silicon revision level.</td></tr></table></td></tr>
<tr><td>111:96</td><td>Vendor ID</td><td>This field contains the PCI-SIG vendor ID for the Management Endpoint.</td></tr>
<tr><td>95:80</td><td>Device ID</td><td>This field contains a vendor assigned device ID for the Management Endpoint.</td></tr>
<tr><td>79:64</td><td>Interface</td><td>This field defines the SMBus version and the Interface Protocols supported.

<table><tr><th>Bits</th><th>Description</th></tr><tr><td>15:8</td><td>Reserved</td></tr><tr><td>7</td><td>ZONE. This field shall be cleared to ‘0’.</td></tr><tr><td>6</td><td>IPMI. This field shall be cleared to ‘0’.</td></tr><tr><td>5</td><td>ASF. This field shall be set to ‘1’. Refer to the MCTP transport binding specification.</td></tr><tr><td>4</td><td>OEM. This field shall be set to ‘1'.</td></tr><tr><td>3:0</td><td>SMBus Version. This field shall be set to 4h for SMBus Version 2.0, or to 5h for SMBus Version 3.0 and 3.1~~which corresponds to SMBus Version 2.0 and 3.0 respectively~~.</td></tr></table></td></tr>
<tr><td>63:48</td><td>Subsystem Vendor ID</td><td>This field contains the PCI-SIG vendor ID for the Management Endpoint.</td></tr>
<tr><td>47:32</td><td>Subsystem Device ID</td><td>This field contains a vendor assigned device ID for the Management Endpoint.</td></tr>
</table>

<table>
<tr><th>Bits</th><th>Field</th><th>Description</th></tr>
<tr><td>31:0</td><td>Vendor Specific ID</td><td>This field ensures all UDIDs from a vendor are unique and is used to associate elements implemented within an NVMe Storage Device. ~~contains a unique 30-bit static NVM storage device ID and distinguish the NVM Subsystem UDID from the FRU Information Device UDID.~~
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:30</td><td>UDID Type. This field distinguishes which NVM subsystem that implements multiple SMBus elements is providing the UDID. Note that Management Controllers implemented prior to NVMe-MI1.1 may be incompatible with devices using values 1h and 3h.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>FRU Information Device</td></tr>
<tr><td>1h</td><td>SMBus/I2C Mux</td></tr>
<tr><td>2h</td><td>Management Endpoint</td></tr>
<tr><td>3h</td><td>Additional devices</td></tr>
</table></td></tr>
<tr><td>~~31~~</td><td>~~**UDID Type.** This field is used to distinguish the Management Endpoint UDID from the VPD UDID. A '1' in this field indicates the Management Endpoint. A '0' in this field indicates the FRU Information Device.~~</td></tr>
<tr><td>~~30~~</td><td>~~Reserved.~~</td></tr>
<tr><td>29:0</td><td>Unique NVM Storage Device ID: This field contains a unique vendor assigned ID for the NVM Subsystem. The ID is different in each NVM Subsystem instance and remains static during the life of the device.</td></tr>
</table></td></tr>
</table>

Modify Section 9.1 as shown below:

9 Management Architecture

9.1 Operational Times

The ability of a Management Endpoint to receive and process Request Messages outlined in this specification is dependent on the state of the Management Endpoint. This section enumerates Management Endpoint operational times and the operations supported in each of these operational times. The NVM Subsystem power state is defined by the state of main power and auxiliary power. Main power consists of one or more voltage rails as defined by form factor. When main power consists of multiple voltage rails, main power is considered "on" when power is good on all main voltage rails. Auxiliary power is optionally supported by a form factor and enables SMBus/I2C communications ~~wake-up processing~~ in the absence of main power. Only the Powered On and Powered Off states are applicable ~~Auxiliary power is considered "off"~~ in form factors and platforms that do not support auxiliary power. ~~Figure 4~~Figure 109 defines the power states of a Management Endpoint. Note that auxiliary power is described from the perspective of the NVMe Storage Device and could be provided by any appropriate power rail in a host platform.

~~**Figure 4: NVM Subsystem Power States**~~

~~**Power State**~~	~~**Main Power**~~	~~**Auxiliary Power**~~
~~Powered Off~~	~~Off~~	~~Off~~
~~Auxiliary Power[1]~~	~~Off~~	~~On~~
~~Main Power (Auxillary Power on if defined)~~	~~On~~	~~On~~
~~Main Power with No Auxiliary Power[1]~~	~~On~~	~~Off~~

~~[1] These states are not used on form factors that do not define Auxiliary power.~~

The operations supported in each NVM ~~S~~subsystem power state are summarized in Figure 5. VPD SMBus/I2C access consists of processing read operations to the FRU Information Device. SMBus/I2C MCTP access consists of processing and responding to MCTP messages and responding to the NVMe Basic Management Command (refer to Appendix A) on the NVM Subsystem SMBus/I2C port. PCIe MCTP access consists of processing and responding to MCTP messages issued on any NVM Subsystem PCIe port. The behavior of an operation that is "Not Supported" in Figure 5 is undefined.

Figure 5: Operations Supported During NVM Subsystem Power States

Operation	**Powered Off** -All Power Rails Off	**Powered On**~~Main Power (with Auxiliary Power)~~ -All Power Rails On	**Auxiliary Power Only**[2] -Main Power Off -Auxiliary Power On	**Main Power Only**[2]~~with No Auxiliary Power~~ -Main Power On -Auxiliary Power Off
SMBus/I2C VPD and SMBus/I2C Mux Access	Not Supported	Supported	Supported	I~~m~~plementation Specific
SMBus/I2C MCTP Access	Not Supported	Supported	Optional[1]	~~Supported~~ Implementation Specific
PCIe MCTP Access	Not Supported	Supported	Not Supported	Supported

NOTES:
1. An implementation that supports SMBus/I2C MCTP Access during Auxiliary Power may support a subset of commands during this power state. The commands that are supported are implementation specific.
2. Auxiliary Power Only and Main Power Only columns are not applicable to form factors that do not define Auxiliary power.

Modify first paragraphs of Section 9.2 as shown below:

9.2 Vital Product Data

The Vital Product Data (VPD) is information describing an NVMe Storage Device. Each NVMe Storage Device ~~Subsystem with one or more Responders~~ shall have a FRU Information Device with a size of 256 bytes to hold the VPD ~~which is compliant with~~as defined in the IPMI Platform Management FRU Information Storage Definition. The VPD shall contain the required elements defined in **Error! Reference source not found.**. ~~The size of the VPD is 256 bytes as defined by the IPMI Platform Management FRU Information Storage Definition.~~

Figure 6**: VPD Elements**

Byte	Name
7:0	Common Header
Vendor Specific	Product Info Area (Optional)
Vendor Specific	MultiRecord Info Area
Vendor Specific	Internal Use Area (Optional)
Vendor Specific	Chassis Info Area (Optional)
Vendor Specific	Board Info Area (Optional)

The VPD shall be accessible using the VPD Read command. The entire contents of the VPD may be updated using the VPD Write command.

If the NVMe Storage Device has an SMBus/I2C interface, the VPD shall be accessible at the SMBus/I2C address of the FRU Information Device using the access mechanism over I2C as defined in the IPMI Platform Management FRU Information Storage Definition. Updating the VPD by writing to the FRU Information Device directly on SMBus/I2C shall not be supported if the VPD Write command is supported.

~~VPD records utilize the Type/Length byte format defined in the IPMI Platform Management FRU Information Storage Definition. Type/Length byte encodings utilized in this specification are summarized in Figure TBD7.~~

Move Figure 141 into section 9.2.2 and add one sentence:

9.2.2 Product Info Area (offset 8 bytes)

The optional Product Info Area shall have the same format and conventions as the Product Info Area Format as defined by the IPMI Platform Management FRU Information Storage Definition. Therefore, all fields within the Product Info Area shall not follow the conventions defined in Section **Error! Reference source not found.**. The Product Info Area factory default values shall be set to the values defined in Figure 8. The Type/Length bytes use the format shown in Figure TBD141a.

Figure TBD7a: Type/Length Byte Format

Bits	Field Name	Description
7:6	Type Code	Specifies field encoding 11b – Always corresponds to ASCII in this specification
5:0	Number of Data Bytes	Specifies field length 000000b indicates that the field is empty

Figure 8: Product Info Area Factory Default Values

Factory Default	Description
01h	**IPMI Format Version Number (IPMIVER):** This field indicates the IPMI Format Version.
Impl Spec	**Product Info Area Length (PALEN):** This field indicates the length of the Product Info Area in multiples of 8 bytes.
19h	**Language Code (LCODE):** This field indicates the language used. A value of 19h is used to indicate English.
~~C8h~~ Impl Spec	**Manufacturer Name Type/Length (MNTL):** This field indicates the type and length of the Manufacturer Name field. The maximum length is 8.
Impl Spec	**Manufacturer Name (MNAME):** This field indicates the Manufacturer name in 8-bit ASCII. Unused bytes should be NULL characters. The Manufacturer name in this field should correspond to that in the PCI Subsystem Vendor ID (SSVID) and IEEE OUI Identifier fields in the Identify Controller Data Structure.
~~D8h~~ Impl Spec	**Product Name Type/Length (PNTL):** This field indicates the type and length of the Product Name field. The maximum length is 24.
Impl Spec	**Product Name (PNAME):** This field indicates the Product name in 8-bit ASCII. Unused bytes should be NULL characters.

Figure 8: Product Info Area Factory Default Values

Factory Default	Description
~~E8h~~ Impl Spec	**Product Part/Model Number Type/Length (PPMNNTL):** This field indicates the type and length of the Product Part/Model Number field. The maximum length is 40.
Impl Spec	**Product Part/Model Number (PPMN):** This field indicates the Product Part/Model Number in 8-bit ASCII. Unused bytes should be NULL characters. This field should contain the same value as the Model Number (NM) field in the NVMe Identify Controller Data Structure.
~~C2h~~ Impl Spec	**Product Version Type/Length (PVTL):** This field indicates the type and length of the Product Part/Model Number field. The maximum length is 2.
Impl Spec	**Product Version (PVER):** This field indicates the Product Version in 8-bit ASCII. Unused bytes should be NULL characters.
~~D4h~~ Impl Spec	**Product Serial Number Type/Length (PSNTL):** This field indicates the type and length of the Product Serial Number field. The maximum length is 20.
Impl Spec	**Product Serial Number (PSN):** This field indicates the Product Serial Number in 8-bit ASCII. Unused bytes should be NULL characters. This field should contain the same value as the Serial Number (SN) field in the NVMe Identify Controller Data Structure.
Impl Spec	**Asset Tag Type/Length (ATTL):** This field indicates the type and length of the Asset Tag field. A value of 00h may be used to indicate an Asset Tag is not present.
Impl Spec	**Asset Tag (AT):** This field indicates the asset tag.
Impl Spec	**FRU File ID Type/Length (ATTL):** This field indicates the type and length of the FRU File ID field. A value of 00h may be used to indicate an FRU File ID is not present.
Impl Spec	**FRU File ID (FFI):** This field provides manufacturing aid for verifying the file that was used during manufacture or field update to load the FRU information.
Impl Spec	**Custom Product Info Area (CPIA):** This optional field allows for the addition of custom Product Info Area fields that shall be proceeded with a Type/Length field.
C1h	**End of Record (EOR):** A value of C1h in this field indicates the end of record.
0h	Zero or more bytes of value 0h that are used to pad the size of the Product Info Area to a multiple of 8 bytes.
Impl Spec	**Product Info Area (PICHK):** Checksum computed over all bytes in the Product Info Area excluding this field. The checksum is computed by adding the 8-bit value of the byes modulo 256 and then taking the 2's complement of this sum. When the checksum and the sum of the bytes module 256 are added, the result should be 0h.

Modify Section 9.2.3 as shown below:

9.2.3 NVMe MultiRecord Area

This MultiRecord is used to describe the form factor, power requirements, and capacity of NVMe Storage Devices with a single NVM Subsystem. It has been superseded by the Topology MultiRecord (refer to Section 9.2.5). For backwards compatibility the NVMe MultiRecord should be included in the VPD unless the NVMe Storage Device has Expansion Connectors, has more than one NVM Subsystem, or if including this MultiRecord would extend the size of the VPD beyond the 256-byte limit.

Figure 9: NVMe MultiRecord Area

Byte Offset	Factory Default	Description
00	0Bh	**NVMe Record Type ID**
01	2h or 82h	~~Bit 7 – end of list; record format version = 2h~~ **Record Format:** Bit | Definition

<table>
<tr><td></td><td></td><td><table>
<tr><td>7</td><td>Set to 1 if last record in list</td></tr>
<tr><td>6:0</td><td>Record format version = 2</td></tr>
</table></td></tr>
<tr><td>02</td><td>3Bh</td><td>Record Length (RLEN): This field indicates the length of the MultiRecord Area in bytes without including the first 5 bytes that are common to all MultiRecords.</td></tr>
<tr><td>03</td><td>Impl Spec</td><td>Record Checksum: This field is used to give the record data a zero checksum (i.e., the modulo 256 sum of the record data bytes from byte offset 05 through the end of this record plus this checksum byte equals zero)</td></tr>
<tr><td>04</td><td>Impl Spec</td><td>Header Checksum: This field is used to give the record header a zero checksum (i.e., the modulo 256 sum of ~~the preceding record bytes starting with~~ the first byte of the header ~~plus~~ through this checksum byte equals zero).</td></tr>
<tr><td>05</td><td>0h</td><td>NVMe MultiRecord Area Version Number: This field indicates the version number of this NVMe MultiRecord. This field shall be set to 0h in this version of the specification.</td></tr>
<tr><td>06</td><td>Impl Spec</td><td>Management Endpoint Form Factor (MEFF): This field indicates the form factor of the Management Endpoint. See the values in Figure 14a.

Editor’s Note: Move Table to Section 9.2.5.2 Figure 14a and updated definition for Value 1

<table>
<tr><th>~~Value~~</th><th>~~Definition~~</th></tr>
<tr><td>~~0~~</td><td>~~Other – unknown~~</td></tr>
<tr><td>~~1 – 15~~</td><td>~~Reserved~~</td></tr>
<tr><td>~~16~~</td><td>~~2.5” Form Factor – unknown~~</td></tr>
<tr><td>~~17~~</td><td>~~2.5” Form Factor – U.2 (SFF-8639) 15mm~~</td></tr>
<tr><td>~~18~~</td><td>~~2.5” Form Factor – U.2 (SFF-8639) 7mm~~</td></tr>
<tr><td>~~19~~</td><td>~~2.5” Form Factor – (SFF-TA-1001) 15mm~~</td></tr>
<tr><td>~~20~~</td><td>~~2.5” Form Factor – (SFF-TA-1001) 7mm~~</td></tr>
<tr><td>~~21 – 31~~</td><td>~~Reserved~~</td></tr>
<tr><td>~~32~~</td><td>~~CEM add in card – unknown~~</td></tr>
<tr><td>~~33~~</td><td>~~CEM add in card – Low Profile (HHHL)~~</td></tr>
<tr><td>~~34~~</td><td>~~CEM add in card – Standard Height Half Length (FHHL)~~</td></tr>
<tr><td>~~35~~</td><td>~~CEM add in card – Standard Height Full Length (FHFL)~~</td></tr>
<tr><td>~~36-47~~</td><td>~~Reserved~~</td></tr>
<tr><td>~~48~~</td><td>~~M.2 module – unknown~~</td></tr>
<tr><td>~~49~~</td><td>~~M.2 module – 2230~~</td></tr>
<tr><td>~~50~~</td><td>~~M.2 module – 2242~~</td></tr>
<tr><td>~~51~~</td><td>~~M.2 module – 2260~~</td></tr>
<tr><td>~~52~~</td><td>~~M.2 module – 2280~~</td></tr>
<tr><td>~~53~~</td><td>~~M.2 module – 22110~~</td></tr>
<tr><td>~~54-63~~</td><td>~~Reserved~~</td></tr>
<tr><td>~~64~~</td><td>~~BGA SSD – unknown~~</td></tr>
<tr><td>~~65~~</td><td>~~BGA SSD – 16 x 20mm (M.2 Type 1620)~~</td></tr>
<tr><td>~~66~~</td><td>~~BGA SSD – 11.5 x 13mm (M.2 Type 1113)~~</td></tr>
<tr><td>~~65-79~~</td><td>~~Reserved~~</td></tr>
<tr><td>~~80~~</td><td>~~Enterprise & Datacenter SSD Form Factor – unknown~~</td></tr>
<tr><td>~~81~~</td><td>~~1U Short Form Factor - (SFF-TA-1006) 5.9mm~~</td></tr>
<tr><td>~~82~~</td><td>~~1U Short Form Factor - (SFF-TA-1006) 8mm~~</td></tr>
<tr><td>~~83~~</td><td>~~1U Long Form Factor - (SFF-TA-1007) 9.5mm~~</td></tr>
<tr><td>~~84~~</td><td>~~1U Long Form Factor - (SFF-TA-1007) 18mm~~</td></tr>
<tr><td>~~85~~</td><td>~~3” Short Form Factor - (SFF-TA-1008) 7.5mm~~</td></tr>
<tr><td>~~86~~</td><td>~~3” Short Form Factor - (SFF-TA-1008) 16.8mm~~</td></tr>
<tr><td>~~87~~</td><td>~~3” Long Form Factor - (SFF-TA-1008) 7.5mm~~</td></tr>
<tr><td>~~88~~</td><td>~~3” Long Form Factor - (SFF-TA-1008) 16.8mm~~</td></tr>
</table></td></tr>
</table>

		~~89-239~~ ~~Reserved~~ ~~240-255~~ ~~Vendor Specific~~
12:07	00h	Reserved
13	Impl Spec[1]	**Initial 1.8V Power Supply Requirements:** This field specifies the initial 1.8V power supply requirements in Watts prior to receiving a Set Slot Power message.
14	Impl Spec[1]	**Maximum 1.8V Power Supply Requirements:** This field specifies the maximum 1.8V power supply requirements in Watts. ~~A value of zero indicates that the power supply voltage is not used.~~
15	Impl Spec[1]	**Initial 3.3V Power Supply Requirements:** This field specifies the initial 3.3V power supply requirements in Watts prior to receiving a Set Slot Power message.
16	Impl Spec[1]	**Maximum 3.3V Power Supply Requirements:** This field specifies the maximum 3.3V power supply requirements in Watts. ~~A value of zero indicates that the power supply voltage is not used.~~
17	00h	Reserved
18	Impl Spec[1]	**Maximum 3.3V aux Power Supply Requirements:** This field specifies the maximum 3.3V power supply requirements in 10 mW units. ~~A value of zero indicates that the power supply voltage is not used.~~
19	Impl Spec[1]	**Initial 5V Power Supply Requirements:** This field specifies the initial 5V power supply requirements in Watts prior to receiving a Set Slot Power message.
20	Impl Spec[1]	**Maximum 5V Power Supply Requirements:** This field specifies the maximum 5V power supply requirements in Watts. ~~A value of zero indicates that the power supply voltage is not used.~~
21	Impl Spec[1]	**Initial 12V Power Supply Requirements:** This field specifies the initial 12V power supply requirements in Watts prior to receiving a Set Slot Power message.
22	Impl Spec[1]	**Maximum 12V Power Supply Requirements:** This field specifies the maximum 12V power supply requirements in Watts. ~~A value of zero indicates that the power supply voltage is not used.~~
23	Impl Spec	**Maximum Thermal Load:** This field specifies the maximum thermal load from the NVM Subsystem in Watts.
36:24	Impl Spec	**Total NVM Capacity:** This field indicates the total NVM capacity of the NVM Subsystem in bytes. If the NVM Subsystem supports Namespace Management, then this field should correspond to the value reported in the TNVMCAP field in the NVMe Identify Controller Data structure. A value of 0h may be used to indicate this feature is not supported.
63:37	00h	Reserved

Notes:

1. Power supply requirements shall be set to the smallest integer value which fully supplies the necessary power to the NVMe Storage Device. A value of 0h indicates that the power supply voltage is not used.

Modify Section 9.2.4 and Figure 145 as shown below, fix font size to 10 for first 2 columns and rows 5-10 of Fig 145 :

9.2.4 NVMe PCIe Port MultiRecord Area

This MultiRecord is used to describe the PCIe connectivity for NVMe Storage Devices with a single NVM Subsystem. It has been superseded by the Topology MultiRecord (refer to Section 9.2.5). For backwards compatibility the PCIe Port MultiRecord should be included in the VPD unless the NVMe Storage Device has Expansion Connectors, has more than one NVM Subsystem, or if including this MultiRecord would extend the size of the VPD beyond the 256-byte limit.

Figure 10: NVMe PCIe Port MultiRecord Area

<table>
<tr><th>Byte Offset</th><th>Factory Default</th><th>Description</th></tr>
<tr><td>00</td><td>0Ch</td><td>NVMe PCIe Port Record Type ID</td></tr>
<tr><td>01</td><td>2h
or
82h</td><td><s>Bit 7 – end of list; record format version = 2h</s>
Record Format:
<table>
<tr><th>Bit</th><th>Definition</th></tr>
<tr><td>7</td><td>Set to 1 if last record in list</td></tr>
<tr><td>6:0</td><td>Record format version = 2</td></tr>
</table></td></tr>
<tr><td>02</td><td>0Bh</td><td>Record Length (RLEN): This field indicates the length of the MultiRecord Area in bytes without including the first 5 bytes that are common to all MultiRecords..</td></tr>
<tr><td>03</td><td>Impl Spec</td><td>Record Checksum: This field is used to give the record data a zero checksum (i.e., the modulo 256 sum of the record data bytes from byte offset 05 to the end of this record plus this checksum byte equals zero).</td></tr>
<tr><td>04</td><td>Impl Spec</td><td>Header Checksum: This field is used to give the record header a zero checksum (i.e., the modulo 256 sum of <s>the preceding record bytes starting with</s> the first byte of the header <s>plus</s> through this checksum byte equals zero).</td></tr>
<tr><td>05</td><td>0h</td><td>NVMe PCIe Port MultiRecord Area Version Number: This field indicates the version number of this NVMe PCIe Port MultiRecord. This field shall be cleared to 0h in this version of the specification.</td></tr>
<tr><td>06</td><td>Impl Spec</td><td>PCIe Port Number: This field contains the PCIe port number. This is the same value as that reported in the Port Number field in the PCIe Link Capabilities Register.</td></tr>
<tr><td>07</td><td>Impl Spec</td><td>Port Information: This field indicates information about the PCIe Ports in the device.

Bits 7:1 are reserved.

Bit 0, if set to ‘1’ indicates that all PCIe ports within the device have the same capabilities (i.e., the capabilities listed in this structure are consistent across each PCIe port).</td></tr>
<tr><td>08</td><td>Impl Spec</td><td>PCIe Link Speed: This field indicates a bit vector of link speeds supported by the PCIe port.
<table>
<tr><th>Bit</th><th>Definition</th></tr>
<tr><td>7:4</td><td>Reserved</td></tr>
<tr><td>3</td><td>Set to ‘1’ if the PCIe link supports 16.0 GT/s. Otherwise cleared to ‘0’.</td></tr>
<tr><td>2</td><td>Set to ‘1’ if the PCIe link supports 8.0 GT/s. Otherwise cleared to ‘0’.</td></tr>
<tr><td>1</td><td>Set to ‘1’ if the PCIe link supports 5.0 GT/s. Otherwise cleared to ‘0’.</td></tr>
<tr><td>0</td><td>Set to ‘1’ if the PCIe link supports 2.5 GT/s. Otherwise cleared to ‘0’.</td></tr>
</table></td></tr>
<tr><td>09</td><td>Impl Spec</td><td>PCIe Maximum Link Width: The maximum PCIe link width for this NVM Subsystem port. This is the expected negotiated link width that the port link trains to if the platform supports it. A Management Controller may compare this value with the PCIe Negotiated Link Width to determine if there has been a PCIe link training issue.</td></tr>
</table>

Figure 10: NVMe PCIe Port MultiRecord Area

<table>
<tr><th>Byte Offset</th><th>Factory Default</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>PCIe x1</td></tr>
<tr><td>2</td><td>PCIe x2</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
<tr><td>4</td><td>PCIe x4</td></tr>
<tr><td>5 to 7</td><td>Reserved</td></tr>
<tr><td>8</td><td>PCIe x8</td></tr>
<tr><td>9 to 11</td><td>Reserved</td></tr>
<tr><td>12</td><td>PCIe x12</td></tr>
<tr><td>13 to 15</td><td>Reserved</td></tr>
<tr><td>16</td><td>PCIe x16</td></tr>
<tr><td>17 to 31</td><td>Reserved</td></tr>
<tr><td>32</td><td>PCIe x32</td></tr>
<tr><td>33 to 255</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>10</td><td>Impl Spec</td><td>MCTP Support: This field contains a bit vector that specifies the level of support for the NVMe Management Interface.

Bits 7:1 are reserved.

Bit 0, if set to ‘1’ indicates that MCTP based management commands are supported on the PCIe port.</td></tr>
<tr><td>11</td><td>Impl Spec</td><td>Ref Clk Capability: This field contains a bit vector that specifies the PCIe clocking modes supported by the port.
<table>
<tr><th>Bit</th><th>Definition</th></tr>
<tr><td>7:4</td><td>Reserved</td></tr>
<tr><td>3</td><td>Set to ‘1’ if the device automatically uses RefClk if provided and otherwise uses SRIS, otherwise cleared to ‘0’.</td></tr>
<tr><td>2</td><td>Set to ‘1’ if the PCIe link supports Separate ReClk with SSC (SRIS), otherwise cleared to ‘0’.</td></tr>
<tr><td>1</td><td>Set to ‘1’ if the PCIe link supports Separate ReClk with no SSC (SRNS), otherwise cleared to ‘0’.</td></tr>
<tr><td>0</td><td>Set to ‘1’ if the PCIe link supports common ReClk, otherwise cleared to ‘0’.</td></tr>
</table>
</td></tr>
<tr><td>15:12</td><td>00h</td><td>Reserved</td></tr>
</table>

Insert new Section 9.2.5 as shown below:

9.2.5 Topology MultiRecord Area

This MultiRecord describes an NVMe Storage Device’s architectural elements and their connections. It is required on all NVMe Storage Devices.

The Topology MultiRecord consists mainly of a list of Element Descriptors as shown in Figure TBD11. Element Descriptors are used to describe the architectural elements that make up an NVMe Storage Device such as NVM subsystems, Upstream Connectors, Expansion Connectors, SMBus/I2C elements, and PCIe elements. Each architectural element has an Element Descriptor Type. The format of an

Element Descriptor is shown in Figure TBD 11b and Element Descriptor Types are listed in Figure TBD12.

Element Descriptors may have fields that are used to point to other Element Descriptors. When an Element Descriptor contains a pointer to another Element Descriptor, then the Element Descriptor containing the pointer is called the parent and the Element Descriptor pointed to by the parent is called the child. An Element Descriptor may be both a child and a parent.

An Element Descriptor pointer is either populated with an index of the child or 0h to indicate that there is no child. The index is a logical construct that indicates the position of an Element Descriptor in the VPD. The Element Descriptor at the lowest byte offset in the VPD has an index of 0, the Element Descriptor at the second lowest byte offset has an index of 1, and so on. A child may have an index that is higher or lower than its parent. The Element Descriptor at the lowest byte offset (i.e., index 0) shall be an Upstream Connector Element Descriptor. Some Element Descriptors use indexes in a similar manner to select a Port from a list of Ports.

Figure TBD11: Topology MultiRecord

<table>
<tr><th>Byte Offset</th><th>Factory Default</th><th>Description</th></tr>
<tr><td>00</td><td>0Dh</td><td>Topology Record Type ID</td></tr>
<tr><td>01</td><td>2h
or
82h</td><td>Record Format:
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7</td><td>Set to 1 if last record in list</td></tr>
<tr><td>6:0</td><td>Record format version = 2</td></tr>
</table></td></tr>
<tr><td>02</td><td>Impl Spec</td><td>Record Length (RLEN): This field indicates the length of the MultiRecord Area in bytes without including the first 5 bytes that are common to all MultiRecords.</td></tr>
<tr><td>03</td><td>Impl Spec</td><td>Record Checksum: This field is used to give the record data a zero checksum (i.e., the modulo 256 sum of the record data bytes from byte offset 05 through the end of this record plus this checksum byte equals zero)</td></tr>
<tr><td>04</td><td>Impl Spec</td><td>Header Checksum: This field is used to give the record header a zero checksum (i.e., the modulo 256 sum of the first byte of the header through this checksum byte equals zero).</td></tr>
<tr><td>05</td><td>00h</td><td>Version Number: This field indicates the version number of this Topology MultiRecord. This field shall be 0h in this version of the specification.</td></tr>
<tr><td>06</td><td>00h</td><td>Reserved</td></tr>
<tr><td>07</td><td>Impl Spec</td><td>Element Count (N): This field indicates the number of Element Descriptors in this Topology MultiRecord. The value of 0h is reserved.</td></tr>
<tr><td>Impl Spec</td><td>Impl Spec</td><td>Element Descriptor 0: This field contains the first Element Descriptor in this Topology MultiRecord.</td></tr>
<tr><td>Impl Spec</td><td>Impl Spec</td><td>Element Descriptor 1: This field contains the second Element Descriptor in this Topology MultiRecord if Element Count is greater than one otherwise this field is not present.</td></tr>
<tr><td>…</td><td>…</td><td>…</td></tr>
</table>

The VPD may contain more than one Topology MultiRecord only when the list of required Element Descriptors is too large to fit into a single Topology MultiRecord. If there is more than one Topology MultiRecord, then the index associated with Element Descriptors continues to increment sequentially across Topology MultiRecord instances. Figure TBD11a illustrates multiple Topology MultiRecords where Index 0 is at the lowest byte offset of any Element Descriptor in the VPD. Parent Element Descriptors may be in different Topology MultiRecords from their Child Element Descriptors.

Figure TBD11a: Indexing Across Extended MultiRecords

Index	Topology MultiRecord Instance	Element Descriptors	Child Indices
0	0	Element Descriptor 0, parent of 2, 3, 5	2, 3, 5
1		Element Descriptor 1, child of 5	
2		Element Descriptor 2, child of 0	
3		Element Descriptor 3, child of 0	
4	1	Element Descriptor 0[1]	
5		Element Descriptor 1, child of 0, parent of 1	1

NOTES:
1. This Element Descriptor is an Extended Element Descriptor that extends the preceding Element Descriptor at index 3. Extended Element Descriptors are further detailed in Section 9.2.5.1.

Figure TBD11b: Element Descriptor

Byte Offset	Factory Default	Description
0	Impl Spec	**Type:** This field indicates the type of the Element Descriptor. Values are defined in Table TBD12
1	Impl Spec	**Revision:** This field indicates the revision of the Element Descriptor.
2	Impl Spec	**Length:** Number of bytes in the Element Descriptor.
Length - 1:3	Impl Spec	This area contains the Type-specific information associated with the Element Descriptor. Type-specific information is defined for each Element Descriptor Type in the subsections below.

Element Descriptor Types, fields, and bits in the VPD that are defined as reserved should be ignored by Requesters to ensure forward and backward compatibility. Extra trailing bytes in an Element Descriptor should be treated as reserved in order to tolerate the Length of an Element Descriptor increasing as new fields are appended in future revisions of the Element Descriptor.

Element Descriptor Types are defined in Figure TBD12. Subsequent sections define the details associated with each Element Descriptor Type.

Figure TBD12: Element Descriptor Types

Value	Name	Reference
0	Reserved	n/a
1	**Extended Element Descriptor**	Section 9.2.5.1
2	**Upstream Connector Element Descriptor**	Section 9.2.5.2
3	**Expansion Connector Element Descriptor**	Section 9.2.5.3
4	**Label Element Descriptor**	Section 9.2.5.4
5	**SMBus/I2C Mux Element Descriptor**	Section 9.2.5.5
6	**PCIe Switch Element Descriptor**	Section 9.2.5.6
7	**NVM Subsystem Element Descriptor**	Section 9.2.5.7
8 to 239	Reserved	n/a
240 to 255	**Vendor specific**	Section 9.2.5.8

9.2.5.1 Extended Element Descriptor

The Extended Element Descriptor is shown in Figure TBD13. This Element Descriptor Type shall only be used when an Element Descriptor spans across more than one Topology MultiRecord. Extended Element Descriptors shall not be the children of other Element Descriptors.

If an Element Descriptor causes the maximum size of a Topology MultiRecord to be exceeded, then that Element Descriptor is truncated so that the non-truncated portion of the Element Descriptor fits into the Topology MultiRecord. The truncated portion of the Element Descriptor forms the contents of the Extended Content field in an Extended Element Descriptor. That Extended Element Descriptor is the first Element Descriptor in the next Topology MultiRecord. If the truncated portion of the Element Descriptor does not fit into a single Topology MultiRecord, then two or more Extended Element Descriptors are required, each in subsequent Topology MultiRecords.

An example is shown in Figure TBD11a where the Element Descriptor at index 4 is an Extended Element Descriptor that extends the Element Descriptor at index 3. Element Descriptor 3 is the child of Element Descriptor 0 and Element Descriptor 4 is not the child of any parent Element Descriptor.

Figure TBD13: Extended Element Descriptor

Byte Offset	Factory Default	Description
00	01h	**Type:** This field indicates the type of the Element Descriptor. The Extended Element Descriptor Type is 1h.
01	00h	**Revision:** This field indicates the revision of the Element Descriptor. The Extended Element Descriptor Revision is 0h for this specification.
02	Impl Spec	**Length:** This field indicates the length of the Extended Element Descriptor in bytes.
Length - 1:03	Impl Spec	**Extended Content:** This field extends the content of the Element Descriptor at the immediately preceding index.

9.2.5.2 Upstream Connector Element Descriptor

The Upstream Connector Element Descriptor is shown in Figure TBD14 and is used to describe an Upstream Connector (i.e., a connector through which a host or Management Controller communicates with the NVMe Storage Device). Upstream Element Descriptors are always a parent and never a child.

Figure TBD14: Upstream Connector Element Descriptor

Byte Offset	Factory Default	Description
00	02h	**Type:** This field indicates the type of the Element Descriptor. The Upstream Connector Element Descriptor Type is 2h.
01	00h	**Revision:** This field indicates the revision of the Element Descriptor. The Upstream Connector Element Descriptor Revision is 0h for this specification.
02	Impl Spec	**Length:** This field indicates the length of the entire Upstream Connector Element Descriptor in bytes.
03	Impl Spec	**Form Factor:** This field indicates the Form Factor of the NVMe Storage Device. See Figure TBD14a for a list of defined values.
04	Impl Spec	**Label Pointer:** If the Upstream Connector has a label, then this field shall contain the index of a Label Element Descriptor that contains the label. The value 0h indicates there is no associated label.
06:05	00h	Reserved

07	Impl Spec	**Maximum Auxiliary Power:** This field specifies the maximum auxiliary power supply requirements in 10 mW increments consumed by the NVMe Storage Device. A value of 0h indicates that auxiliary power is not used from this Upstream Connector.
09:08	Impl Spec	**Maximum Power:** This field specifies the maximum power in Watts consumed by the NVMe Storage Device.
10	Impl Spec	**Upstream Port Descriptor Count:** This field indicates the number of Upstream Port Descriptors associated with this Upstream Connector Element Descriptor. The permitted range of values is 1 to 64.
Impl Spec	Impl Spec	**Upstream Port Descriptor 0:** This field contains the first Upstream Port Descriptor
Impl Spec	Impl Spec	**Upstream Port Descriptor 1:** This field contains the second Upstream Port Descriptor in this Upstream Connector Element Descriptor if Port Descriptor Count is greater than one otherwise this field is not present.
…	…	…

The value of the Form Factor field indicates the NVMe Storage Device's form factor. Figure TBD14a lists the NVMe Storage Device's Form Factor values.

Figure TBD14a: Form Factors

Value	Description
0	Other – unknown
1	Integrated
2 to 15	Reserved
16	2.5" Form Factor – unknown
17	2.5" Form Factor – U.2 (SFF-8639) 15mm
18	2.5" Form Factor – U.2 (SFF-8639) 7mm
19	2.5" Form Factor – (SFF-TA-1001) 15mm
20	2.5" Form Factor – (SFF-TA-1001) 7mm
21 to 31	Reserved
32	CEM add in card – unknown
33	CEM add in card – Low Profile (HHHL)
34	CEM add in card – Standard Height Half Length (FHHL)
35	CEM add in card – Standard Height Full Length (FHFL)
36 to 47	Reserved
48	M.2 module – unknown
49	M.2 module – 2230
50	M.2 module – 2242
51	M.2 module – 2260
52	M.2 module – 2280
53	M.2 module – 22110
54 to 63	Reserved
64	BGA SSD – unknown
65	BGA SSD – 16 x 20mm (M.2 Type 1620)
66	BGA SSD – 11.5 x 13mm (M.2 Type 1113)
65 to 79	Reserved
80	Enterprise & Datacenter SSD Form Factor – unknown
81	1U Short Form Factor - (SFF-TA-1006) 5.9mm
82	1U Short Form Factor - (SFF-TA-1006) 8mm

83	1U Long Form Factor - (SFF-TA-1007) 9.5mm
84	1U Long Form Factor - (SFF-TA-1007) 18mm
85	3” Short Form Factor - (SFF-TA-1008) 7.5mm
86	3” Short Form Factor - (SFF-TA-1008) 16.8mm
87	3” Long Form Factor - (SFF-TA-1008) 7.5mm
88	3” Long Form Factor - (SFF-TA-1008) 16.8mm
89 to 239	Reserved
240 to 255	Vendor Specific

The Upstream Connector may have an associated label, such as silk screened text on the printed circuit board. If the Upstream Connector has a label, then the Label Pointer may contain the index of the associated Label Element Descriptor.

The Upstream Connector Element Descriptor contains a list of the Upstream Port Descriptors that are ports through which a host or Management Controller communicates with the NVMe Storage Device. Each Upstream Port Descriptor has a type. The types defined in this specification are SMBus/I2C Upstream Port Descriptor and PCIe Upstream Port Descriptor.

An SMBus/I2C Upstream Port Descriptor is shown in Figure TBD14b. It contains a list of pointers to child Element Descriptors whose SMBus/I2C port is directly connected to the Upstream Connector.

A PCIe Upstream Port Descriptor is shown in Figure TBD14c. It indicates the starting and ending PCIe lane numbers on the Upstream Connector that make up a PCIe Upstream Port. The PCIe Upstream Port Descriptor contains a single pointer to a child Element Descriptor connected to this PCIe Upstream Port. The Destination Port field of the PCIe Upstream Port Element Descriptor specifies which port of the child is connected to this Upstream Connector. The Destination Port value is an index into the child Element Descriptor’s list of Port Descriptors.

The PCIe lanes associated with a PCIe Upstream Connector may be organized as a single large port or subdivided into multiple ports. Each of these ports is described with its own PCIe Upstream Port Descriptor. The PCIe Upstream Port Descriptors may be listed in any order. A form factor specific mechanism, such as the U.2 Dual Port Enable signal, may be used to determine which of the listed PCIe Upstream Port Descriptors are currently applicable. These form factor specific mechanisms are outside the scope of this specification

For example, a U.2 NVMe Storage Device capable of running in either single-port mode or dual-port mode based on the Dual Port Enable signal would have three PCIe Upstream Port Descriptors describing PCIe ports on the following PCIe Lanes:

1. PCIe lanes 0 to 3 (single-port mode)
2. PCIe lanes 0 to 1 (dual-port mode)
3. PCIe lanes 2 to 3 (dual-port mode)

In the example above, if the U.2 NVMe Storage Device is only capable of running in single-port mode, then only the PCIe Upstream Port Descriptor describing the single-port mode (item 1 in the list above) shall be included in the Upstream Connector Element Descriptor. And if the U.2 NVMe Storage Device is only capable of running in dual-port mode, then only the two PCIe Upstream Port Descriptors describing the dual-port mode (items 2 and 3 in the list above) shall be included in the Upstream Connector Element Descriptor.

In another example, consider a x16 CEM add-in card Upstream Connector that is subdivided into four x4 PCIe ports, also referred to as bifurcation. Each of these x4 PCIe Upstream Ports may connect to different elements on the NVMe Storage Device. The Upstream Connector in this example shall contain four PCIe Upstream Port Descriptors describing the four PCIe ports:

1. PCIe lanes 0 to 3
2. PCIe lanes 4 to 7

3. PCIe lanes 8 to 11
4. PCIe lanes 12 to 15

Figure TBD14b: SMBus/I2c Upstream Port Descriptor

Byte Offset	Factory Default	Description
00	00h	**Type:** This field indicates the type of the Port Descriptor. The SMBus/I2C Port Descriptor Type is 0h.
01	Impl Spec	**Length:** This field indicates the length of the SMBus/I2C Port Descriptor in bytes.
02	Impl Spec	**Count:** This field indicates the number of SMBus/I2C Pointers in the SMBus/I2C Upstream Port Descriptor. The permitted range of values is 1 to 32.
03	Impl Spec	**SMBus/I2C Pointer 0:** This field contains the child index of the first Element Descriptor whose SMBus/I2C port is connected to this SMBus/I2C port.
04	Impl Spec	**SMBus/I2C Pointer 1:** If Count is greater than one, then this field is present and contains the child index of the second Element Descriptor whose SMBus/I2C port is connected to this SMBus/I2C Upstream Port. If Count is not greater than one, then this field is not present.
…	…	…

Figure TBD14c: PCIe Upstream Port Descriptor

Byte Offset	Factory Default	Description
00	01h	**Type:** This field indicates the type of Upstream Port Descriptor. The PCIe Upstream Port Descriptor Type is 1h.
01	Impl Spec	**Length:** This field indicates the length of the PCIe Upstream Port Descriptor in bytes.
02	Impl Spec	**Starting Lane:** This field indicates first PCIe lane (i.e., lane 0) of the port from the Upstream Connector.
03	Impl Spec	**Ending Lane:** This field indicates the ending PCIe lane of the port from the Upstream Connector.
04	Impl Spec	**PCIe Pointer:** This field contains the child index of the Element Descriptor whose PCIe port is connected to this PCIe Upstream Port.
05	Impl Spec	**Destination Port:** This field contains the index of the Port Descriptor in the child Element Descriptor. If the child Element Descriptor has one PCIe upstream port (i.e., a PCIe Switch Element Descriptor) this field shall be cleared to 0h.

9.2.5.3 Expansion Connector Element Descriptor

The Expansion Connector Element Descriptor is shown in Figure TBD17 and is used to describe the form factor, label, and port configurations for Expansion Connectors on a Carrier. The Expansion Connector Element Descriptor shall be a child Element Descriptor.

Figure TBD17: Expansion Connector Element Descriptor

Byte Offset	Factory Default	Description

00	03h	**Type:** This field indicates the type of the Element Descriptor. The Expansion Connector Element Descriptor Type is 3h.
01	00h	**Revision:** This field indicates the revision of the Element Descriptor. The Expansion Connector Element Descriptor Revision is 0h for this specification.
02	Impl Spec	**Length:** This field indicates the length of the Expansion Connector Element Descriptor in bytes.
03	Impl Spec	**Form Factor:** This field indicates the Form Factor of the NVMe Storage Device that plugs into the Expansion Connector. See Figure TBD14a for a list of defined values.
04	Impl Spec	**Label Pointer:** If the Upstream Connector has a label, then this field shall contain the index of a Label Element Descriptor that contains the label. The value 0h indicates there is no associated label.
05	Impl Spec	**Expansion Connector Port Descriptor Count:** This field indicates the number of Expansion Port Descriptors associated with this Expansion Connector Element Descriptor. The permitted range of values is 1 to 64.
06	Impl Spec	**Expansion Connector Port Descriptor 0:** This field contains the first Expansion Connector Port Descriptor.
07	Impl Spec	**Expansion Connector Port Descriptor 1:** This field contains the second Expansion Connector Port Descriptor in this Expansion Connector Descriptor if Expansion Connector Port Descriptor Count is greater than one otherwise this field is not present.
…	…	…

In a manner similar to the PCIe Upstream Connector, the Expansion Connector Element Descriptor's PCIe lanes may support one or more PCIe ports for connecting to external NVMe Storage Device FRUs. The PCIe ports have a starting and ending PCIe lane number on the Expansion Connector that are determined by the external NVMe Storage Device FRU's form factor's lane numbering.

The Expansion Connector Element Descriptor holds the list of Expansion Connector PCIe Port Descriptors. Each PCIe port is described by an Expansion Connector PCIe Port Descriptor whose format is shown in Figure TBD17a. Parent Element Descriptors, such as Upstream Connectors and PCIe Switches, contain Port Descriptors that point to Expansion Connectors. The Destination Port field of the parent Port Descriptor contains an index to the specific Expansion Connector PCIe Port Descriptor instance to which the Port Descriptor is connected. Each Expansion Connector PCIe Port Descriptor is the destination of exactly one pointer from a parent Element Descriptor.

There are two ways to document Expansion Connector's on Carriers with permanently populated expansion devices. If the FRU Information Device attached to the Upstream Connector of the Carrier can be programmed from an NVM Subsystem, then the permanently attached expansion device should be described as if its elements were directly on the Carrier, this Expansion Connector Element Descriptor is not included in the Carrier's VPD, and the FRU Information Device on the expansion device becomes optional. Otherwise, the Form Factor field for the Expansion Connector should be set to Integrated and the Requester will also need to read the VPD from the expansion device which is not modified from its original content by the permanent connection.

Figure TBD17a: Expansion Connector PCIe Port Descriptor

Byte Offset	Factory Default	Description
00	00h	**Type:** This field indicates the type of Expansion Connector Port Descriptor. The Expansion Connector PCIe Port Descriptor Type is 0.
01	Impl Spec	**Length:** This field indicates the length of the Expansion Connector PCIe Port Descriptor in bytes.

02	Impl Spec	**Starting Lane:** This field indicates first PCIe lane (i.e., lane 0) of the port on the Expansion Connector PCIe Port Descriptor.
03	Impl Spec	**Ending Lane:** This field indicates the ending PCIe lane of the port on the Expansion Connector PCIe Port Descriptor.

9.2.5.4 Label Element Descriptor

The Label Element Descriptor is shown in Figure TBD18 and is used to store text strings in the VPD for Element Descriptors that have a label. A Label Element Descriptor shall be a child Element Descriptor.

Figure TBD18: Label Element Descriptor

Byte Offset	Factory Default	Description
00	04h	**Type:** This field indicates the type of the Element Descriptor. The Label Element Descriptor Type is 4h.
01	00h	**Revision:** This field indicates the revision of the Element Descriptor. The Label Element Descriptor Revision is 0h for this specification.
02	Impl Spec	**Length:** This field indicates the length of the Label Element Descriptor in bytes including the null termination.
Length - 1 : 03	Impl Spec	**Label String:** This field contains a null-terminated UTF-8 string used to identify the parent Element Descriptor.

9.2.5.5 SMBus/I2C Mux Element Descriptor

The SMBus/I2C Mux Element Descriptor is shown in Figure TBD19 and is used to describe an SMBus/I2C multiplexor element that can connect a single upstream SMBus/I2C channel to zero or more downstream SMBus/I2C channels. This Element Descriptor contains the address and capabilities of the SMBus/I2C Mux followed by a list of SMBus/I2C Mux Channel Descriptors that describe SMBus/I2C Mux downstream channel connections. The SMBus/I2C Mux shall be compatible with the industry standard PCA9542/45/48 family of SMBus/I2C multiplexors and may be extended to support ARP, error detection, and additional downstream channels as defined below.

Figure TBD19: SMBus/I2C Mux Element Descriptor

<table>
<tr><th>Byte Offset</th><th>Factory Default</th><th>Description</th></tr>
<tr><td>00</td><td>05h</td><td>Type: This field indicates the type of the Element Descriptor. The SMBus/I2C Mux Element Descriptor Type is 5h.</td></tr>
<tr><td>01</td><td>00h</td><td>Revision: This field indicates the revision of the Element Descriptor. The SMBus/I2C Mux Element Descriptor Revision is 0h for this specification.</td></tr>
<tr><td>02</td><td>Impl Spec</td><td>Length: This field indicates the length of the SMBus/I2C Mux Element Descriptor in bytes.</td></tr>
<tr><td>03</td><td>E8h or E9h</td><td>SMBus/I2C Address Info: This field indicates the SMBus/I2C address and whether or not ARP is supported.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7:1</td><td>SMBus/I2C Address: This field contains the 7-bit SMBus/I2C address. Refer to Table TBD3 for requirements.</td></tr>
<tr><td>0</td><td>ARP Capable: This bit is set to ‘1’ if SMBus ARP is supported, else it is cleared to ‘0’. Refer to Table TBD3 for requirements.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>04</td><td>Impl Spec</td><td>SMBus/I2C Capabilities: This field indicates the SMBus/I2C Mux capabilities.

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7</td><td>Form Factor Reset: This field is set to ‘1’ if all of the SMBus/I2C reset mechanisms are supported as defined by the associated form factor specification. This field is cleared to ‘0’ if the form factor does not define SMBus Reset or the NVMe Storage Device does not support all of the SMBus/I2C reset mechanisms defined in the specification for the Form Factor in the Host Connector Element Descriptor.</td></tr>
<tr><td>6</td><td>Packet Error Code (PEC) Support: This field is set to ‘1’ if PEC is supported by the SMBus/I2C Mux. This field is cleared to ‘0’ if PEC is not supported.</td></tr>
<tr><td>6:2</td><td>Reserved</td></tr>
<tr><td>1:0</td><td>Maximum Speed: This field is set to the highest supported SMBus/I2C clock speed by the SMBus/I2C Mux.

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>100 kHz</td></tr>
<tr><td>1</td><td>400 kHz</td></tr>
<tr><td>2</td><td>1 MHz</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
</table></td></tr>
</table></td></tr>
<tr><td>05</td><td>Impl Spec</td><td>SMBus/I2C Mux Channel Descriptor Count: This field indicates the number of downstream channels listed for this SMBus/I2C Mux. Each channel has a corresponding SMBus/I2C Channel Descriptor in the list below. The permitted range of values is 1 to 64. The value of this field may be less than the actual number of Channels implemented by the SMBus/I2C Mux if the truncated SMBus/I2C Mux Channel Descriptors are not connected to anything.</td></tr>
<tr><td>06</td><td>Impl Spec</td><td>SMBus/I2C Mux Channel Descriptor 0: This field contains the first SMBus/I2C Mux Channel Descriptor</td></tr>
<tr><td>07</td><td>Impl Spec</td><td>SMBus/I2C Mux Channel Descriptor 1: This field contains the second SMBus/I2C Mux Channel Descriptor in this SMBus/I2C Mux Element Descriptor if SMBus/I2C Mux Channel Descriptor Count is greater than one otherwise this field is not present.</td></tr>
<tr><td>…</td><td>…</td><td>…</td></tr>
</table>

An SMBus/I2C Mux Channel Descriptor is shown in Figure TBD19b. SMBus/I2C Mux Channel Descriptors that are not connected to anything have a value 0h in the Count field and contain no SMBus/I2C Mux Channel Descriptors. Unconnected SMBus/I2C Mux Channel Descriptors at the end of the list in Figure TBD19 may be truncated unless they are needed to position the optional Packet Error Code (PEC).

Writing to an SMBus/I2C Mux configures the SMBus/I2C Mux and reading from an SMBus Mux returns its current configuration. Figure TBD19a shows the protocol for reading and writing an SMBus/I2C Mux configuration. The white background blocks are transmitted by a Management Controller and the grey background blocks are transmitted in response by the SMBus/I2C Mux. The first byte sent or received is the SMBus/I2C Mux address followed by one or more channel bytes. Each channel byte has eight channel bits that are set to ‘1’ for connecting the corresponding downstream channel to the upstream channel or cleared to ‘0’ for disconnecting the corresponding downstream channel from the upstream channel.

The first channel byte sent or received represents channels 0 to 7, the second channel byte sent or received represents channels 8 to 15, and so on. Within each channel byte the least significant bit in the byte that is transmitted or received represents the lowest numbered channel. Bits for channels exceeding the SMBus/I2C Mux Channel Descriptor Count are reserved.

Figure TBD19a: SMBus/I2C Mux Read and Write Command Format

The minimum number of channel bytes are read or written to reach all the channels specified in the SMBus/I2C Mux Channel Descriptor Count field. Thus, SMBus/I2C Muxes with one to eight downstream channels would have one channel byte while an SMus/I2C Mux with 57 downstream channels would have 8 channel bytes. In the example shown in Figure TBD19a, the SMBus/I2C Mux has 16 downstream channels that require 2 bytes. In this example, channels 1 and 8 are being connected while all others are being disconnected.

An SMBus/I2C Mux may also protect communications with an optional Packet Error Code (PEC) that is appended after sufficient channel bytes have been read or written to satisfy the SMBus/I2C Mux Channel Descriptor Count value. If the write command includes a PEC byte and the PEC byte is incorrect, then the entire command shall be ignored by the SMBus/I2C Mux. Otherwise, the actions associated with the write command take place after the STOP condition is received. Write commands with insufficient channel bytes shall be accepted with truncated channel bytes having an implied value of zero. Bytes beyond the size needed for the number of channels and PEC are reserved.

Multiple downstream channels may be simultaneously connected to the upstream channel to bridge them together. All downstream channels shall be disconnected when the NVMe Storage Device is powered off (see Figure 109) or by an SMBus Reset (refer to Section 9.3.4). Connecting or disconnecting channels while they are active is strongly discouraged and results in undefined behavior.

Figure TBD19b: SMBus/I2C Mux Channel Descriptor

Byte Offset	Factory Default	Description
00	00h	**Type:** This field indicates the type of the Descriptor. The SMBus/I2C Mux Channel Descriptor Type is 0h.
01	Impl Spec	**Length:** This field indicates the length of the SMBus/I2C Mux Channel Descriptor in bytes.
02	Impl Spec	**Count:** This field indicates the number of SMBus/I2C Pointers in the SMBus/I2C Mux Channel Descriptor. The permitted range of values is 0 to 32.
03	Impl Spec	**SMBus/I2C Pointer 0:** This field contains the child index of the first Element Descriptor whose SMBus/I2C is connected to this channel.
04	Impl Spec	**SMBus/I2C Pointer 1:** If Count is greater than one, then this field is present and contains the child index of another Element Descriptor whose SMBus/I2C is connected to this channel. If Count is not greater than one, then this field is not present.
…	…	…

9.2.5.6 PCIe Switch Element Descriptor

The PCIe Switch Element Descriptor is shown in Figure TBD21 and is used to describe a PCIe switch. This Element Descriptor is the child of a single parent and the parent of one or more children.

Figure TBD21: PCIe Switch Element Descriptor

Byte Offset	Factory Default	Description
00	06h	**Type:** This field indicates the type of the Element Descriptor. The PCIe Switch Element Descriptor Type is 6h.
01	00h	**Revision:** This field indicates the revision of the Element Descriptor. The PCIe Switch Element Descriptor Revision is 0h for this specification.
02	Impl Spec	**Length:** This field indicates the length of the PCIe Switch Element Descriptor in bytes.
03	Impl Spec	**Upstream Switch Port Descriptor:** This field contains the PCIe Switch Port Descriptor that describes the upstream switch port.
Impl Spec	Impl Spec	**Downstream Switch Port Descriptor Count:** This field indicates the number of PCIe Port Descriptors associated with downstream switch ports.
Impl Spec	Impl Spec	**Downstream Switch Port Descriptor 0:** This field contains the PCIe Switch Port Descriptor associated with the first downstream port.
Impl Spec	Impl Spec	**Downstream Switch Port Descriptor 1:** This field contains the PCIe Switch Port Descriptor associated with the second downstream port if Downstream Switch Port Descriptor Count is greater than one otherwise this field is not present.
…	…	…

The PCIe Switch Element Descriptor consists of a list of PCIe Switch Port Descriptors. There is an Upstream Switch Port Descriptor that describes the upstream port and is the child of exactly one parent Element Descriptor. A variable length list of Downstream Switch Port Descriptors describe the downstream ports.

The format of a PCIe Switch Port Descriptor is shown in Figure TBD21a. It describes the PCIe port's supported PCIe link speeds, PCIe maximum link width, reference clock capabilities, and PCIe Port Number. Downstream ports also have a child Element Descriptor and its Destination Port index value.

Figure TBD21a: PCIe Switch Port Descriptor

Byte Offset	Factory Default	Description
00	00h	**Type:** This field indicates the type of Port Descriptor. The PCIe Switch Port Descriptor Type is 0.
01	Impl Spec	**Length:** This field indicates the length of the PCIe Switch Port Descriptor in bytes.
02	Impl Spec	**PCIe Link Speed:** This field indicates a bit vector of link speeds supported by the PCIe port.

Bit	Description
7:4	Reserved
3	Set to '1' if the PCIe link supports 16 GT/s. Otherwise cleared to '0'.
2	Set to '1' if the PCIe link supports 8.0 GT/s. Otherwise cleared to '0'.
1	Set to '1' if the PCIe link supports 5.0 GT/s. Otherwise cleared to '0'.

<table>
<tr><td></td><td></td><td><table><tr><td>0</td><td>Set to ‘1’ if the PCIe link supports 2.5 GT/s. Otherwise cleared to ‘0’.</td></tr></table></td></tr>
<tr><td>03</td><td></td><td>PCIe Maximum Link Width: The maximum PCIe link width for this port.
<table><tr><th>Value</th><th>Definition</th></tr><tr><td>0</td><td>Reserved</td></tr><tr><td>1</td><td>PCIe x1</td></tr><tr><td>2</td><td>PCIe x2</td></tr><tr><td>3</td><td>Reserved</td></tr><tr><td>4</td><td>PCIe x4</td></tr><tr><td>5 to 7</td><td>Reserved</td></tr><tr><td>8</td><td>PCIe x8</td></tr><tr><td>9 to 11</td><td>Reserved</td></tr><tr><td>12</td><td>PCIe x12</td></tr><tr><td>13 to 15</td><td>Reserved</td></tr><tr><td>16</td><td>PCIe x16</td></tr><tr><td>17 to 31</td><td>Reserved</td></tr><tr><td>32</td><td>PCIe x32</td></tr><tr><td>33 to 255</td><td>Reserved</td></tr></table></td></tr>
<tr><td>04</td><td>Impl Spec</td><td>RefClk Capability: This field contains a bit vector that specifies the PCIe clocking modes supported by the port.
<table><tr><th>Bit</th><th>Description</th></tr><tr><td>7:4</td><td>Reserved</td></tr><tr><td>3</td><td>Set to ‘1’ for upstream ports that automatically use RefClk if provided and otherwise uses SRIS. Otherwise, cleared to ‘0’. Reserved for downstream ports.</td></tr><tr><td>2</td><td>Set to ‘1’ if the PCIe port supports Separate RefClk with SSC (SRIS), otherwise cleared to ‘0’.</td></tr><tr><td>1</td><td>Set to ‘1’ if the PCIe port supports Separate RefClk with no SSC (SRNS), otherwise cleared to ‘0’.</td></tr><tr><td>0</td><td>Set to ‘1’ if the PCIe port supports common RefClk, otherwise cleared to ‘0’.</td></tr></table></td></tr>
<tr><td>05</td><td>Impl Spec</td><td>Port Number: This field indicates the PCIe Port Number, as defined by the PCI Express Base Specification, associated with this port.</td></tr>
<tr><td>06</td><td>Impl Spec</td><td>PCIe Pointer: In downstream ports this field contains the child index of the Element Descriptor that has a PCIe port connected to this PCIe port. In upstream ports this field is cleared to 0h.</td></tr>
<tr><td>07</td><td>Impl Spec</td><td>Destination Port: This field contains the index of the Port Descriptor in the child Element Descriptor. If the child Element Descriptor has one PCIe upstream port (i.e., a PCIe Switch Element Descriptor) this field shall be cleared to 0h.</td></tr>
</table>

9.2.5.7 NVM Subsystem Element Descriptor

The NVM Subsystem Element Descriptor is shown in Figure TBD23 and is used to describe an NVM Subsystem contained in the NVMe Storage Device.

Figure TBD23: NVM Subsystem Element Descriptor

Byte Offset	Factory Default	Description

<table>
<tr><td>00</td><td>07h</td><td>Type: This field indicates the type of the Element Descriptor. The NVM Subsystem Element Descriptor Type is 7h.</td></tr>
<tr><td>01</td><td>00h</td><td>Revision: This field indicates the revision of the Element Descriptor. The NVM Subsystem Element Descriptor Revision is 0h for this specification.</td></tr>
<tr><td>02</td><td>Impl Spec</td><td>Length: This field indicates the length of the NVM Subsystem Element Descriptor in bytes.</td></tr>
<tr><td>03</td><td>3Ah or 3Bh</td><td>SMBus/I2C Address Info: If the NVM Subsystem supports an MCTP over SMBus/I2C port, then this field indicates the SMBus/I2C address for MCTP over SMBus/I2C port and whether or not SMBus ARP is supported; otherwise, this field has a value of 0h.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7:1</td><td>SMBus/I2C Address: This field contains the 7-bit SMBus/I2C address. Refer to Table TBD3 for requirements.</td></tr>
<tr><td>0</td><td>ARP Capable: This bit is set to ‘1’ if SMBus ARP is supported, else it is cleared to ‘0’. Refer to Table TBD3 for requirements.</td></tr>
</table></td></tr>
<tr><td>04</td><td>Impl Spec</td><td>SMBus/I2C Capabilities: If the NVM Subsystem supports an SMBus/I2C port then this field indicates the SMBus/I2C capabilities; otherwise, this field has a value of 0h.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7</td><td>Reset: This field is set to ‘1’ if all of the SMBus/I2C reset mechanisms are supported as defined by the associated form factor specification. This field is cleared to ‘0’ if the form factor does not define SMBus Reset or the NVMe Storage Device does not support all of the SMBus/I2C reset mechanisms defined by the specification for the Form Factor in the Host Connector Element Descriptor.</td></tr>
<tr><td>6:2</td><td>Reserved</td></tr>
<tr><td>1:0</td><td>Maximum Speed: This field is set to the highest supported SMBus/I2C clock speed.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>100 kHz</td></tr>
<tr><td>1</td><td>400 kHz</td></tr>
<tr><td>2</td><td>1 MHz</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
</table></td></tr>
</table></td></tr>
<tr><td>05</td><td>Impl Spec</td><td>NVM Subsystem Port Descriptor Count: This field indicates the number of NVM Subsystem Port Descriptors associated with the NVM Subsystem. The permitted range of values is 1 to 64.</td></tr>
<tr><td>Impl Spec</td><td>Impl Spec</td><td>NVM Subsystem Port Descriptor 0: This field contains the NVM Subsystem Port Descriptor associated with the first NVM Subsystem port.</td></tr>
<tr><td>Impl Spec</td><td>Impl Spec</td><td>NVM Subsystem Port Descriptor 1: This field contains the NVM Subsystem Port Descriptor associated with the second NVM Subsystem port if NVM Subsystem Port Descriptor Count is greater than one otherwise this field is not present.</td></tr>
<tr><td>…</td><td>…</td><td>…</td></tr>
</table>

Each upstream port is described by an NVM Subsystem Port Descriptor as shown in Figure TBD23a. It describes the PCIe port’s supported PCIe link speeds, PCIe max link width, RefClk capabilities, and PCIe Port Identifier. Each NVM Subsystem Port Descriptor should be the child of exactly parent Element Descriptor.

Figure TBD23a: NVM Subsystem Port Descriptor

<table>
<tr><th>Byte Offset</th><th>Factory Default</th><th>Description</th></tr>
<tr><td>00</td><td>00h</td><td>Type: This field indicates the type of an NVM Subsystem Port Descriptor. The NVM Subsystem Port Descriptor Type is 0.</td></tr>
<tr><td>01</td><td>Impl Spec</td><td>Length: This field indicates the length of the NVM Subsystem Port Descriptor in bytes.</td></tr>
<tr><td>02</td><td>Impl Spec</td><td>PCIe Link Speed: This field indicates a bit vector of link speeds supported by the PCIe port.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7:4</td><td>Reserved</td></tr>
<tr><td>3</td><td>Set to ‘1’ if the PCIe link supports 16 GT/s. Otherwise cleared to ‘0’.</td></tr>
<tr><td>2</td><td>Set to ‘1’ if the PCIe link supports 8.0 GT/s. Otherwise cleared to ‘0’.</td></tr>
<tr><td>1</td><td>Set to ‘1’ if the PCIe link supports 5.0 GT/s. Otherwise cleared to ‘0’.</td></tr>
<tr><td>0</td><td>Set to ‘1’ if the PCIe link supports 2.5 GT/s. Otherwise cleared to ‘0’.</td></tr>
</table></td></tr>
<tr><td>03</td><td></td><td>PCIe Maximum Link Width: The maximum PCIe link width for this NVM Subsystem port.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>PCIe x1</td></tr>
<tr><td>2</td><td>PCIe x2</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
<tr><td>4</td><td>PCIe x4</td></tr>
<tr><td>5 to 7</td><td>Reserved</td></tr>
<tr><td>8</td><td>PCIe x8</td></tr>
<tr><td>9 to 11</td><td>Reserved</td></tr>
<tr><td>12</td><td>PCIe x12</td></tr>
<tr><td>13 to 15</td><td>Reserved</td></tr>
<tr><td>16</td><td>PCIe x16</td></tr>
<tr><td>17 to 31</td><td>Reserved</td></tr>
<tr><td>32</td><td>PCIe x32</td></tr>
<tr><td>33 to 255</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>04</td><td>Impl Spec</td><td>RefClk Capability: This field contains a bit vector that specifies the PCIe clocking modes supported by the port.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7:4</td><td>Reserved</td></tr>
<tr><td>3</td><td>Set to ‘1’ if the device automatically uses RefClk if provided and otherwise uses SRIS, otherwise cleared to ‘0’.</td></tr>
<tr><td>2</td><td>Set to ‘1’ if the PCIe link supports Separate RefClk with SSC (SRIS), otherwise cleared to ‘0’.</td></tr>
<tr><td>1</td><td>Set to ‘1’ if the PCIe link supports Separate RefClk with no SSC (SRNS), otherwise cleared to ‘0’.</td></tr>
<tr><td>0</td><td>Set to ‘1’ if the PCIe link supports common RefClk, otherwise cleared to ‘0’.</td></tr>
</table></td></tr>
<tr><td>05</td><td>Impl Spec</td><td>Port Identifier: This field contains the NVMe-MI Port Identifier associated with this port.</td></tr>
</table>

9.2.5.8 Vendor-Specific Element Descriptors

The Vendor-Specific Element Descriptor is shown Figure TBD24.

Figure TBD24: Vendor-Specific Element Descriptors

Byte Offset	Factory Default	Description
00	Impl Spec	**Type:** This field indicates the type of the Element Descriptor. Vendor-Specific Types have a value in the range of F0h – FFh.
01	Impl Spec	**Revision:** This field indicates the revision of the Element Descriptor. The Vendor-Specific Element Descriptor Revision is determined by the Vendor.
02	Impl Spec	**Length:** This field indicates the length of the Vendor-Specific Element Descriptor in bytes.
04:03	Impl Spec	**PCI Vendor ID:** This field indicates PCI-SIG assigned vendor identifier.
Impl Spec	Impl Spec	**Vendor Specific:** Vendor-specific information.

Add Section 9.3.4 as shown below:

9.3.4 SMBus Reset

All SMBus/I2C elements should support the recommendation for SMBus reset when the SMBus/I2C clock is low for longer than $t_{TIMEOUT,MIN}$.

Some form factors may also specify one or more separate SMBus reset mechanisms. If such mechanisms are supported by an NVMe Storage Device, then the NVMe Storage Device shall propagate the reset to all SMBus/I2C elements on the NVMe Storage Device and translate the reset as needed to Expansion Connector form factors.

If the SMBus/I2C element on an NVMe Storage Device is in master mode, then an SMBus Reset shall cause it to generate a STOP condition as defined in the SMBus Specification within or after the current data byte in the transfer process. The NVMe Storage Device shall remain idle on SMBus for the remainder of the SMBus Reset assertion even if other SMBus/I2C elements attempt to address it. An NVMe Storage Device shall be ready to receive a START condition as defined in the SMBus Specification within 10 ms after SMBus Reset de-assertion.

An SMBus Reset shall not modify ARP assigned addresses. Management Controllers may send an ARP reset after the SMBus Reset if addresses need to be reinitialized.

An SMBus Reset shall cause SMBus/I2C Management Endpoints to drop the MCTP packet in flight. If the MCTP Command Servicing State is in Transmit then it shall change to Idle as if transmit completed. An SMBus Reset does not reset other MCTP state information or abort NVMe-MI command processing.

Change Appendix A as shown below:

The SMBus slave address to read this data structure is not the same address ~~we~~ used for MCTP, and defaults to 6Ah if ARP is not invoked[1]. After the Management Controller assigns the MCTP UDID a new address using ARP, then the Basic Management Command will respond to slave reads at the MCTP address on NVMe Storage Devices with multiple controllers that support ARP. This method of changing the Basic Command address is optional on other NVMe Storage Device implementations. Since SMBus shifts the address left to make room for the read/write direction bit, the address appears in the examples below as D4h for write and D5h for read.

…

The SMBus Arbitration bit may be used for simple arbitration on systems that have multiple drives on the same SMBus channel~~segment~~ without ARP or muxes to separate them. To use this mechanism, the host follows this 3 step process to handle collisions for the same slave address:

…

Change Figure 146 as shown below:

Command Code	Offset (byte)	Description
0	00	**Length of Status:** Indicates number of additional bytes to read before encountering PEC. This value should always be 6 (06h) in implementations of this version of the spec.
	01	**Status Flags (SFLGS):** This field indicates the status of the NVM Subsystem. **SMBus Arbitration** – Bit 7 is set to '1' after an SMBus block read is completed all the way to the stop bit without bus contention and cleared to '0' if an SMBus Send Byte FFh is received on this SMBus slave address. **Drive Not Ready** – Bit 6 is set to '1' when the subsystem is not capable of processing NVMe management commands, and the rest of the transmission may be invalid. If cleared to '0', then the NVM Subsystem is fully powered and ready to respond to management commands. This logic level intentionally identifies and prioritizes powered up and ready drives over their powered off neighbors on the same SMBus channel~~segment~~. **Drive Functional** – Bit 5 is set to '1' to indicate an NVM Subsystem is functional. If cleared to '0', then there is an unrecoverable failure in the NVM Subsystem and the rest of the transmission may be invalid. Note that this bit may default to '0' after reset and transition to '1' after the NVM Subsystem has completed initialization and this case should not be considered an error. **Reset Not Required** - Bit 4 is set to '1' to indicate the NVM Subsystem does not need a reset to resume normal operation. If cleared to '0', then the NVM Subsystem has experienced an error that prevents continued normal operation. A Controller Level Reset is required to resume normal operation. **Port 0 PCIe Link Active** - Bit 3 is set to '1' to indicate the first port's PCIe link is up (i.e., the Data Link Control and Management State Machine is in the DL_Active state). If cleared to '0', then the PCIe link is down. **Port 1 PCIe Link Active** - Bit 2 is set to '1' to indicate the second port's PCIe link is up. If cleared to '0', then the second port's PCIe link is down or not present. Bits 1:0 shall be set to '1'.
…	…	…
32+	32:255	Vendor Specific – Th~~is~~ese data structures shall not exceed the maximum read length of 255 specified in the SMBus version 3 specification. Preferably their lengths are ~~is~~ not greater than 32 for compatibility with SMBus 2.0~~, additional blocks shall be on 8 byte boundaries~~.