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 - 2018 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 servicemarks may be claimed as the property of their respective owners.

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

NVM Express Technical Proposal for New Feature

Technical Proposal ID	**6001 – SES Based Enclosure Management**
Change Date	**1/28/2018**
Builds on Specification	**NVM Express Management Interface 1.0a**

Technical Proposal Author(s)

Name	Company
Peter Onufryk	Microsemi

This technical proposal defines an enclosure management architecture for NVM Express. This technical proposal also defines the Management Endpoint Buffer which is an intermediate buffer that allows NVMe-MI to service out-of-band NVMe-MI Messages that have a Message Body that is larger than the 4224 byte limit that is specified by the NVMe Management Messages over MCTP Binding Specification.

Revision History

Revision Date	Change Description
11/11/2016	• Initial draft.
12/4/2016	• Corrected typos and refined wording • Removed the Invalid Operation Requested (INVOP) bit and associated functionality • Removed text associated with zero filling of data not transferred by an SES Send command and added text that requires that all bytes of a page be transferred in a series of non-overlapping SES Send commands. • Added Page Changed (PCHANGED) bit to SES Receive command to allow page changes to be detected.
3/18/2017	• Changed Short Enclosure Status diagnostic page support to optional • Remove page code (PCODE) field from SES Send command to better align with SCSI Send Diagnostic command. • Added definitions for Enclosure/NVMe Enclosure and Enclosure Services Process • Started writing the architectural model for an enclosure (Section 1.5)
4/2/2017	• Removed text associated with ad-hoc greater than 4KB transfers for SES Send and SES Receive. • Expanded Section 1.5 Enclosures and Enclosure Management
5/1/2017	• Updated Section 1.5 based on review feedback • Built out sections and commands associated with the MCTP Management Endpoint Buffer
5/7/2017	• Changed all references to Management Endpoint Buffer to MCTP Management Endpoint Buffer. The prefix MCTP was added to highlight the fact that only MCTP Management Endpoints may support this feature. • Added Section 4.6 that describes the MCTP Management Endpoint Buffer and its operation.
5/22/2017	• Integrated Austin's feedback. • Added text that describes that updates to the MCTP Management Endpoint Buffer are not atomic. • Described how the MCTP Management Endpoint Buffer operates when a sanitize operation is performed on the NVM subsystem. • Updated Figure F+2 to show a usage model that is realistic for the case of multiple Enclosure Services Processes associated with a single enclosure.
6/5/2017	• Simplified the SES Send and SES Receive commands by using the MCTP Management Endpoint Buffer. Removed ability to retrieve/transfer a partial page using the SES Send or SES Receive commands.
6/12/2017	• Updated error handling. • Added back NVMe Management Response field to SES Receive command.
6/26/2017	• Added text that SES state in NVMe-MI is global and not per I_T Nexus. • Changed name of MCTP Management Endpoint Buffer to Management Endpoint Buffer. • Added error codes for SES and Management Endpoint Buffer access that reads data zeroed due to a sanitize operation. • Added mapping of SES sense keys and additional sense codes to NVMe-MI Response Message Status values.
7/10/2017	• Added Section 1 edits to comprehend that NVMe-MI now also specifies enclosure management. • Added column to opcode tables to show optional and mandatory commands for an NVMe Storage Device as well as an NVMe Enclosure. • Miscellaneous other updates and corrections.
7/16/2017	• Changed sanitize operation behavior to always clear the entire Management Endpoint Buffer. • Updated text and command tables to allow some commands to be prohibited in an NVMe Enclosure. Described how an NVMe Enclosure that is also an NVMe Storage Device is required to implement mandatory

	commands for either. Described how an NVMe Enclosure that is also an NVMe Storage Device may implement optional NVMe Storage Device commands.
8/14/2017	• Removed table of supported SES control type diagnostic pages and SES status type diagnostic pages and replaced table references with a reference to SES-3. • If applicable, split command tables into multiple tables. If applicable, added separate table that shows commands supported in-band and one that shows commands supported out-of-band. • Put shortened definition terms in parentheses. • Miscellaneous wording and text changes to improve readability. • Added NVM Subsystem Report (NVMSR) field to NVM Subsystem Information Data Structure. This allows one to determine if the NVM subsystem is part of an NVMe Storage Device, and NVMe Enclosure or both.
9/11/2017	• Clarified that the Request Data contains an SES control type diagnostic page. The Length of the Request Data is equal to the value of the PAGE LENGTH field in the SES control type diagnostic page plus four. • Added Allocation length (ALENGTH) field to NVMe Management Dword 1. Described how ALENGTH is used to limit the maximum amount of SES diagnostic page data that may be returned. • Updated TP based on review feedback. • Changed all instances of NVMe-MI MCTP Message to NVMe-MI Message. • Moved NVMe Storage Devices outside of Enclosures and showed them attaching through slots.
9/24/2017	• Corrected ALENGTH field typo/error. • Added Data Length (DLEN) field to SES Send command and updated text. • Modified MEB length error handling to more accurately align with the protocol. • Corrected error handling when MEB is used with a command that doesn't support it. • Incorporated editorial feedback.
9/25/2017	• Capitalized all instances of Subenclosure. Added Subenclosure to the list of definitions. • Incorporated editorial feedback. • Incorporated feedback from the workgroup call review.
10/1/2017	• Moved NVM Subsystem Report (NVMSR) field from NVM Subsystem Information Data Structure to Identify Controller Data Structure. • Changed format of Management Endpoint Capabilities (MEC) field in Identify Controller Data Structure to match other bit field definitions in the NVMe-MI spec. Named bits zero and one. • Changed support of Read NVMe-MI Data Structure when using the In-Band Tunneling Mechanism from mandatory (M) to prohibited (P). This command is no longer mandatory since the NVMSR field was moved to the Identify Controller Data Structure.
10/2/2017	• Moved NVMSR field in Identify Controller Data Structure from byte 254 to 253 since VPD write cycle reduction TP uses byte 253.
10/8/2017	• Incorporated feedback received from workgroup review. • Updated Section 1.4 NVMe Storage Device Architectural Model to clarify that the Subsystem in this case shall include a non-volatile storage medium.

10/22/2017	• Removed NVMe Storage Device definition since that definition will be contained in NVMe 1.3 Alignment TP. Updated architectural model section to use NVMe Storage Device FRU and provide a linkage between an NVMe Storage Device (shorthand) and an NVMe Storage Device FRU. • Changed instance of NVM storage device to NVMe Storage Device.
11/5/2017	• Removed “An example of an NVMe Storage Device is a PCIe SSD” text. • Minor editorial text edits.
11/10/2017	• Modified management interface command support tables to make NVMe-MI commands optional. Added note that states that the mapping of these commands to an NVMe Enclosure is outside the scope of this specification. • Added text to NVM Subsystem Report (NVMSR) field that at least one bit in this field shall be set to ‘1’. • Changed second instance of Table Xb to Xc to fix table labeling error.
1/28/2018	• Editorial changes to align with NVMe TP numbering, updated NVMe administration, and updated year.

Note – This technical proposal alludes to the fact that NVMe-MI may be accessed both in-band and out-of-band since it was developed in parallel with the Support for In-Band NVMe-MI technical proposal. The details of in-band access shall be updated in the Support for In-Band NVMe-MI technical proposal.

Description of Specification Changes

Add the following subsections to Section 1 as shown below:

1 Introduction

1.1 Overview

NVM Express (NVMe) is a register-level interface that allows in-band host software to communicate with an NVM Subsystem. Since this specification builds on the NVM Express specification, knowledge of NVMe is assumed.

The NVMe Management Interface (NVMe-MI) allows a Management Controller to communicate out-of-band with an NVMe NVM Subsystem over one or more external interfaces. NVMe-MI also allows a Management Controller to monitor and control the elements of an NVMe Enclosure.
~~Since this specification builds on the NVMe specification, knowledge of NVMe is assumed.~~

1.2 Scope

This specification defines an architecture and command set for out-of-band management of an ~~NVMe~~ NVM Subsystem~~.~~ as well as an architecture and mechanisms for monitoring and controlling the elements of an NVMe Enclosure.

NVMe-MI has the following key capabilities for NVMe Storage Devices:

- Discover devices that are present and learn capabilities of each device
- Store data about the host environment enabling a Management Controller to query the data later
- Health and temperature monitoring
- Multiple Command Slots to prevent a long latency command from blocking monitoring operations
- Processor and operating system agnostic

- A standard format for VPD and defined mechanisms to read/write VPD contents
- Preserves data at rest security

NVMe-MI has the following key capabilities for NVMe Enclosures:

- Discover enclosures and learn their capabilities
- Manage and sense the state of enclosure power supplies, cooling devices, displays, and indicators
- Discover NVMe Storage Devices that are present in enclosure slots
- Preserves data at rest security

1.2.1 Outside of Scope

The architecture and command set are specified apart from any usage model. This specification does not specify whether NVMe is used to implement a solid-state drive, a main memory, a cache memory, a backup memory, a redundant memory, etc. Specific usage models are outside the scope, optional, and not licensed.

This interface is NVM technology agnostic and is specified at a level that abstracts implementation details associated with any specific NVM technology. For example, NAND wear leveling, block erases, and other management tasks are abstracted.

The implementation or use of other published specifications referred to in this specification, even if required for compliance with the specification, are outside the scope of this specification (e.g., PCI Express, SMBus/I2C and MCTP).

The management of NVMe FRUs containing multiple architecturally visible NVM ~~s~~Subsystems is outside the scope of this specification. This specification does not define new security mechanisms.

This specification does not cover management of non-transparent bridges or management using any interface other than MCTP over PCIe VDM or SMBus/I2C. Co-ordination between multiple Management Controllers or a Management Controller and a device other than a Management Endpoint is outside the scope of this specification.

Coordinating concurrency resulting from operations associated with multiple Management Endpoints or between a host and Management Endpoint operations is outside the scope of this specification.

The specification of specific enclosure elements that make up an NVMe Enclosure is outside the scope of this specification. Support for cards or modules that connect to a device slot element (slot) of an NVMe Enclosure, that are not NVMe Storage Devices (e.g., GPUs or FPGAs) is outside the scope of this specification.

An Enclosure may support comprehensive management capabilities using SCSI Enclosure Services, basic management capabilities using transport specific mechanisms, or no management capabilities. An example of basic enclosure management capabilities is Native PCIe Enclosure Management (NPEM) specified by the PCI-SIG for PCI Express. The specification of such transport specific basic management capabilities is outside the scope of this specification. This specification only defines comprehensive management using SCSI Enclosure Services.

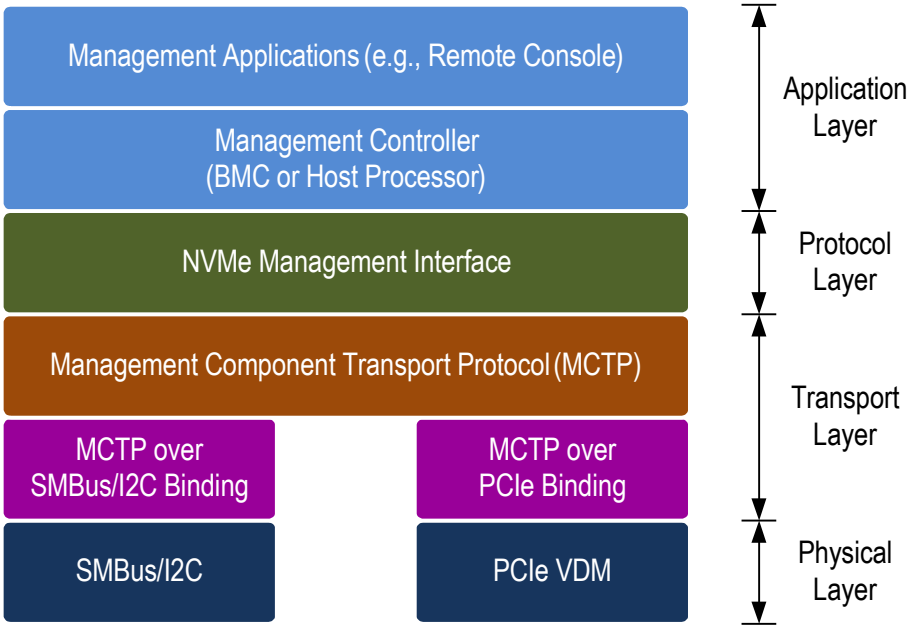
An Enclosure may contain multiple Enclosure Services Processes. Communication and coordination between the Enclosure Services Processes that manage Enclosure elements is outside the scope of this specification.

1.3 ~~Theory of Operation~~ NVMe-MI Out-of-Band Protocol Layering

NVMe-MI is designed to provide a common interface over multiple physical layers (i.e., PCI Express, SMBus/I2C) for inventory, monitoring, configuration, and change management. The interface provides

the flexibility necessary to manage NVM Subsystems using an out-of-band mechanism in a variety of host environments and systems.

Figure 1: NVMe Management Interface Protocol Layering



NVMe-MI utilizes the Management Component Transport Protocol (MCTP) as the command transport and utilizes existing MCTP SMBus/I2C and PCIe bindings for the physical layer. MCTP commands are submitted to one of two Command Slots associated with each Management Endpoint.

Modify the NVMe-MI specification to capitalize all instances of Storage Device or NVMe Storage Device

Modify Section 1.4 as shown below:

1.4 NVMe Storage Device Field Replaceable Unit Architectural Model

An NVMe ~~s~~Storage ~~d~~Device Field Replaceable Unit, or simply NVMe Storage Device~~, such as a PCIe SSD,~~ that implements this specification~~,~~ consists of an NVM Subsystem that includes a non-volatile storage medium along with one or more Management Endpoints. There may be up to one Management Endpoint per PCIe port and 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. The Port Identifier for a PCIe port is the same as the Port Number field in the PCIe Link Capabilities Register.

Insert the following section after Section 1.4 as shown below:

1.5 NVMe Enclosure Architectural Model

An NVMe Enclosure, or Enclosure, is a platform, card, module, box, rack, or set of boxes that may provide power, cooling, and mechanical protection for one or more NVM Subsystems. These NVM

Subsystems may be part of the Enclosure itself and/or may be contained in NVMe Storage Devices that connect to the Enclosure through one or more enclosure slots.

An Enclosure may contain elements that support operation of the Enclosure (e.g., power supplies, fans, locks, temperature sensors, current sensors, and voltage sensors). An Enclosure may also contain displays and/or indicators that indicate the state of the Enclosure (e.g., state of elements, NVM Subsystems, or RAID volumes) and/or NVMe Storage Devices that connect to the Enclosure. Some of the elements that make up an Enclosure may be removable and replaceable while the Enclosure continues to operate normally.

SCSI Enclosure Services - 3 (SES-3) is a standard developed by the American National Standards Institute T10 committee for management of Enclosures using the SCSI architecture. While the NVMe and SCSI architectures differ, the elements of an Enclosure and the capabilities required to manage these elements are similar. Thus, NVMe-MI leverages SES for enclosure management. SES manages the elements of an Enclosure using control and status diagnostic pages transferred using SCSI commands (refer to Enclosure Control and Enclosure Status diagnostic pages in SES-3). NVMe-MI uses these same control and status diagnostic pages, but transfers them using the SES Send and SES Receive commands. Since enclosure management is tightly coupled with NVMe-MI, NVMe-MI supports only a standalone Enclosure Services Process model.

A Management Controller manages an Enclosure using SES Send and SES Receive commands that are part of the Management Interface Command Set (refer to Section 5). The SES Send command provides the functionality of the SES-3 SCSI SEND DIAGNOSTIC command and is used by a Management Controller to send SES control type diagnostic pages to modify the state of the Enclosure. The SES Receive command provides the functionality of the SES-3 SCSI RECEIVE DIAGNOSTIC RESULTS command and is used by a Management Controller to retrieve SES status type diagnostic pages that contain various status and warning information available from the Enclosure.

Refer to SES-3 for a list and description of SES control type diagnostic pages and SES status type diagnostic pages. The NVMe firmware update process is used (i.e., Firmware Image Download and Firmware Commit commands) to update NVMe firmware. Download Microcode Control and Status diagnostic pages, if supported, shall only be supported on Enclosure elements.

An Enclosure Services Process, that is logically part of the Enclosure, is responsible for managing Enclosure elements and participates in servicing SES Send and SES Receive commands issued by a Management Controller. Unlike the SES-3 Enclosure Services Process model that maintains state for each I_T nexus (refer to SES-3), unless otherwise noted, NVMe-MI maintains a single global state for an Enclosure regardless of the Management Controller or path used to access that state.

An Enclosure may contain of one or more Subenclosures (refer to SES-3). Each Subenclosure is identified by a SES-3 defined one-byte Subenclosure identifier. If multiple Subenclosures are present, then one of the Subenclosures is designated as the primary Subenclosure and the remaining Subenclosures are secondary Subenclosures. When an Enclosure consists of only a single Subenclosure, then that Subenclosure is the primary Subenclosure. The Enclosure Services Process associated with the primary Subenclosure is the one that provides access to Enclosure services information for all Subenclosures. Refer to SES-3 for more information.

An Enclosure contains one or more Enclosure slots that are used as a means for a Requester to communicate with the Enclosure and/or as a means for the Enclosure to communicate with an NVMe Storage Device that connects to the Enclosure. Associated with each Enclosure slot is a SES element that may be used to manage the slot. Refer to SES-3 for more information.

Figure F illustrates an example NVMe Enclosure that contains one NVM Subsystem. This Enclosure has multiple ports that Requesters may use to communicate with the Enclosure. It also has multiple slots that are used to connect NVMe Storage Devices to the Enclosure (e.g., PCIe). The mapping of Enclosure

ports to NVM subsystems, NVMe Controllers within these NVM subsystems, and NVMe Storage Devices is vendor specific and outside the scope of this specification. An Enclosure shall contain one or more NVM Subsystems used for enclosure management. The enclosure in this example may be managed using the out-of-band mechanism via the Management Endpoint ("Mgmt. Ep." in the figure), or using the in-band tunneling mechanism via the NVMe Controller.

Figure F: Example NVMe Enclosure

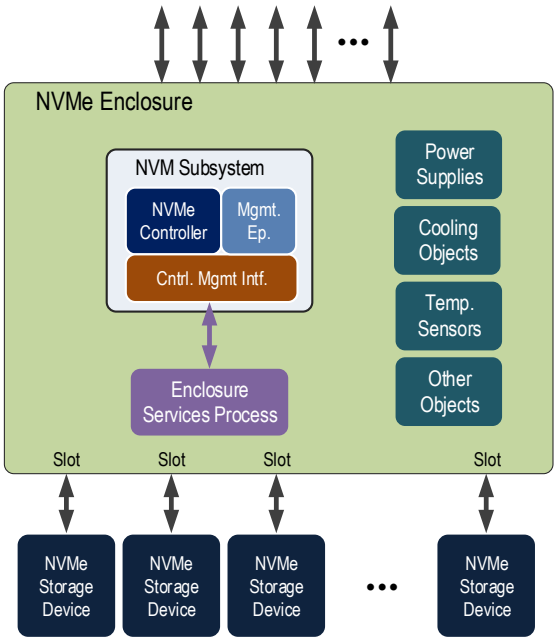


Figure F+1 illustrates an example NVMe Enclosure that contains multiple NVM subsystems and no NVMe Storage Devices. This may represent a software storage appliance. The NVM subsystems and Controllers contained within these NVM subsystems may be real or emulated in software. Not all Controllers within these NVM subsystems need to have the same capabilities. Some of the possible capability configurations are illustrated in this example. Some Controllers in this example simply provide access to namespaces; others provide access to namespaces and in-band NVMe-MI management capabilities; and others provide access to namespaces and a Management Endpoint for out-of-band management capabilities.

Figure F+1: Example NVMe Enclosure with Multiple NVM Subsystems

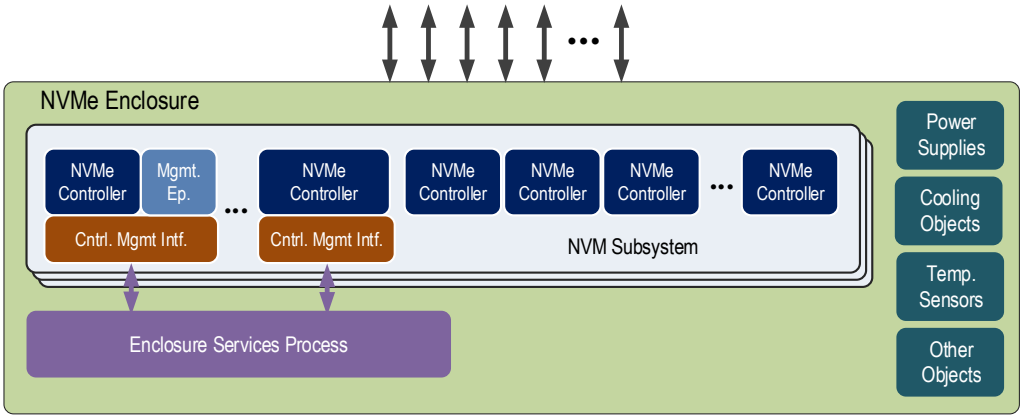


Figure F+2 shows an Enclosure that supports two Enclosure Services Processes. Elements of the Enclosure may be accessible by one or both of these Enclosure Services Processes. The coordination of access to elements by multiple Enclosure Services Processes is outside the scope of this specification.

Figure F+2: Example NVMe Enclosure with Multiple Enclosure Services Processes

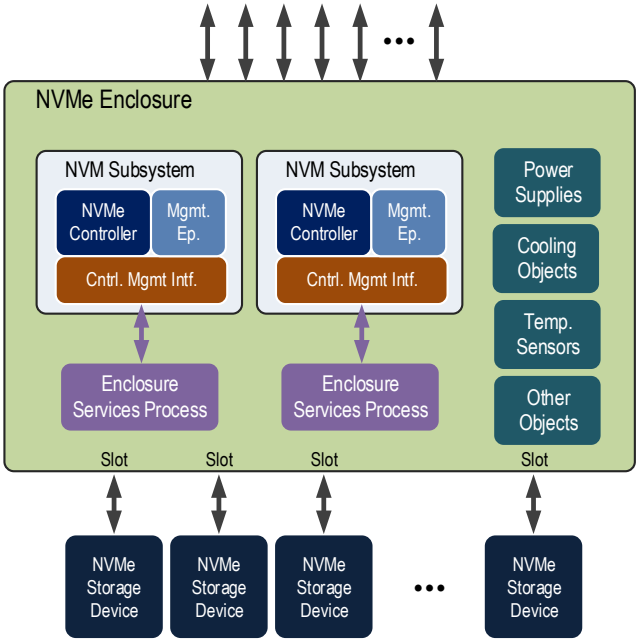
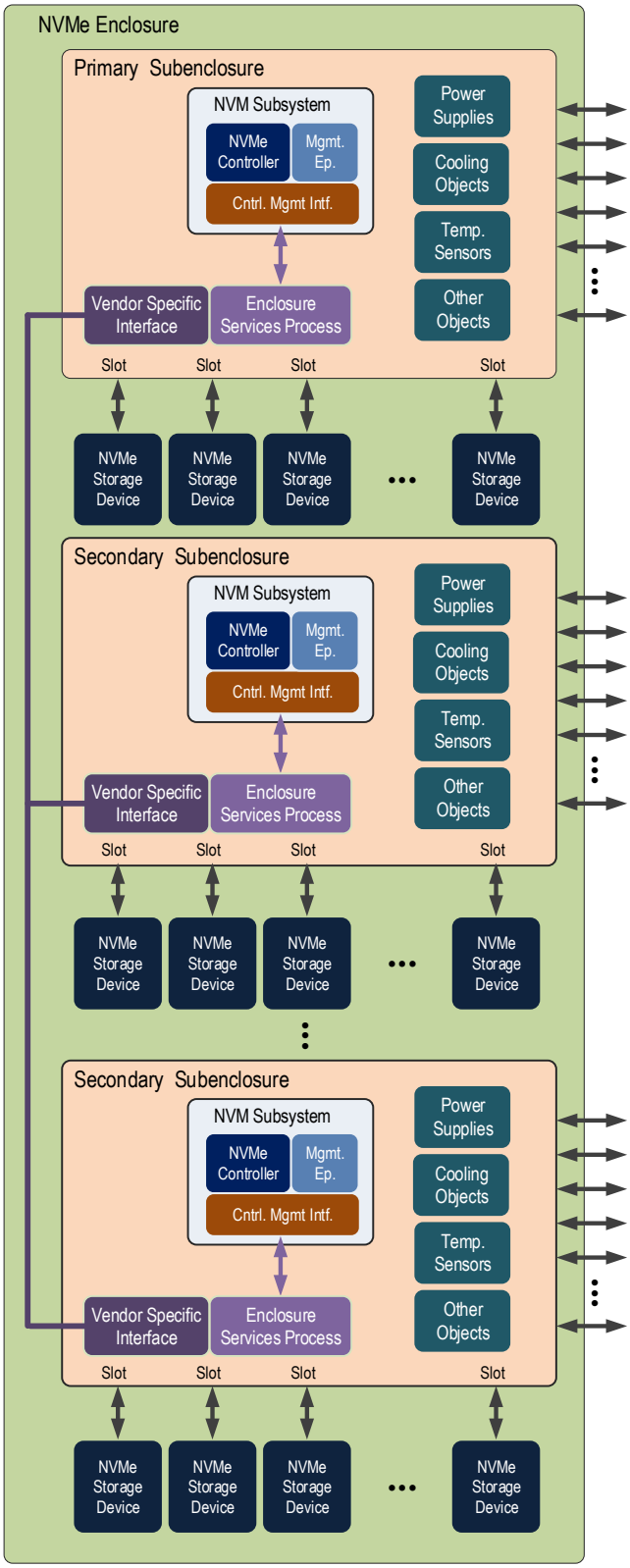


Figure F+3 shows an Enclosure that consists of multiple Subenclosures. Each Subenclosure in this example contains an Enclosure Services Process. Enclosure services information from Subenclosures is combined into a single set of SES diagnostic pages by the primary Subenclosure. A Subenclosure identifier is used to distinguish from which Subenclosure the information was obtained. Refer to SES-3 for more information. A primary Subenclosure may access Enclosure services information in Subenclosures using the out-of-band mechanism, the in-band tunneling mechanism, or both; or may use a vendor specific interface. This example illustrates the use of a vendor specific interface.

Figure F+3: Example NVMe Enclosure with Subenclosures



Certain Enclosure behaviors are managed by setting controls and testing status of elements within an Enclosure. An Enclosure Services Process may also monitor a variety of warning and error conditions. These conditions may be communicated to the Management Controller through polling by the Management Controller (refer to Enclosure Services Management mode page in SES-3 for details).

The mapping of SES-3 sense keys and additional sense codes associated with CHECK CONDITION status to NVMe-MI Response Message Status values is shown in Figure F+4. The asynchronous event notification reporting mechanism described in SES-3 is not supported by NVMe-MI.

Figure F+4: Mapping of SES-3 Sense Keys and Additional Sense Codes to Response Message Status

Reseponse Message	SES-3	
	Sense Key	Additional Sense Code
Enclosure Services Failure	HARDWARE ERROR	ENCLOSURE SERVICES FAILURE
Enclosure Services Transfer Failure		ENCLOSURE SERVICES TRANSFER FAILURE
Enclosure Failure		ENCLOSURE FAILURE
Enclosure Services Transfer Refused	HARDWARE ERROR or ILLEGAL REQUEST	ENCLOSURE SERVICES TRANSFER REFUSED
Unsupported Enclosure Function	ILLEGAL REQUEST	UNSUPPORTED ENCLOSURE FUNCTION
Enclosure Services Unavailable	NOT READY	ENCLOSURE SERVICES UNAVAILABLE
Enclosure Degraded	RECOVERED ERROR	WARNING – ENCLOSURE DEGRADED

Add the following subsections to Section 1.5.1 in alphabetical order (and alphabetize the existing definition):

1.5.1.x1 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.

1.5.1.x2 Enclosure Management

The discovery, monitoring and control of elements that make up an NVMe Enclosure.

1.5.1.y Enclosure Services Process

A process that implements Enclosure services for an NVMe Enclosure that supports enclosure management. Refer to SCSI Enclosure Services - 3 (SES-3) for more information.

1.5.1.z NVMe Subenclosure (Subenclosure)

A portion of an Enclosure accessed through a primary Enclosure's Enclosure Services Process.

1.5.1.a Management Endpoint Buffer

An intermediate buffer that allows NVMe-MI to service out-of-band NVMe-MI Messages that have a Message Body that is larger than the 4224 byte limit that is specified by the NVMe Management Messages over MCTP Binding Specification.

1.5.1.c NVMe Subenclosure (Subenclosure)

A portion of an enclosure accessed through a primary subenclosure's enclosure services process. Refer to SCSI Enclosure Services - 3 (SES-3) for more information.

Add the following reference to Section 1.7 as shown below:

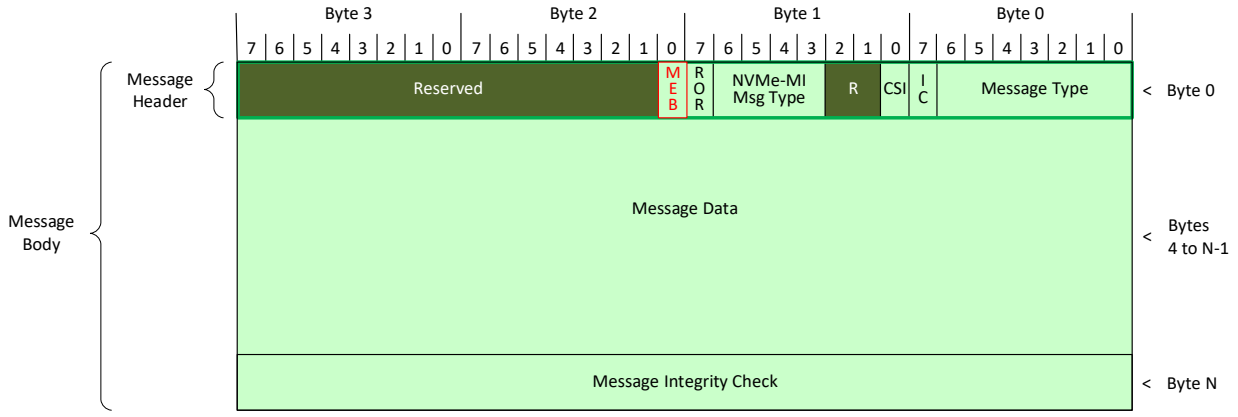
T10/xxxx-D SCSI SCSI Enclosure Services - 3 (SES-3)

Modify Section 3.2 as shown below:

MCTP Messages

An MCTP message consists of the payload of one or more MCTP packets. The maximum sized message is 4224 bytes (4K + 128). Refer to the NVMe Management Messages over MCTP Binding Specification. Messages with lengths greater than 4224 are considered invalid messages. The format of an NVMe-MI ~~MCTP~~ message is shown in Figure 10.

Figure 10: NVMe-MI ~~MCTP~~ Message



Message Fields

The format of an NVMe-MI Message consists of a Message Header in the first Dword, followed by the Message Data, and ends with the Message Integrity Check Dword as shown in Figure 10.

The Message Header contains a Message Type (MT) field and an Integrity Check (IC) field that are defined by the MCTP Base Specification. The Message Type field specifies the type of payload contained in the message body and is required to be set to 4h in all messages associated with NVMe-MI (refer to the MCTP IDs and Codes specification). The Integrity Check (IC) field indicates whether the message is covered by an overall MCTP Message Integrity Check. All NVMe-MI Messages are protected by a 32-bit CRC computed over the message body contents. The IC field shall be set to '1' in all NVMe-MI ~~MCTP~~ messages.

The Request or Response (ROR) bit in the Message Header specifies whether the NVMe-MI ~~MCTP~~ message is associated with a Request Message or a Response Message. The NVMe Message Type (NMIMT) field specifies whether the Request Message is a Control Primitive or a specific type of Command Message (refer to Figure 14). Finally, the Command Slot Identifier (CSI) field specifies the Command Slot with which the message is associated. Refer to section 4 for additional information about Command Slots.

The Management Endpoint Buffer (MEB) bit in the Message Header specifies whether

Request/Response Data is contained in the associated Request/Response Data field of an NVMe-MI Message or in the Management Endpoint Buffer. This bit should only be set in Command Messages that support Management Endpoint Buffer operation (i.e., those listed in the Management Endpoint Buffer Supported Command List data structure). It is an error to set this bit in any other Command Message and when this occurs it causes the Command Message to complete with an Invalid Parameter error status.

Figure 11: NVMe-MI ~~MCTP~~ Message Fields

<table>
<tr><th>Byte</th><th>Description</th></tr>
<tr><td>1</td><td>
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7</td><td>Request or Response (ROR): This field indicates whether the message is a Request Message or Response Message. This field is cleared to ‘0’ for Request Messages. This field is set to ‘1’ for Response Messages.</td></tr>
<tr><td>6:3</td><td>NVMe-MI Message Type (NMIMT): This field specifies the NVMe-MI Message Type.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>Control Primitive – refer to section 4.4</td></tr>
<tr><td>1h</td><td>NVMe-MI Command – refer to section 5</td></tr>
<tr><td>2h</td><td>NVMe Admin Command – refer to section 6</td></tr>
<tr><td>3h</td><td>Reserved</td></tr>
<tr><td>4h</td><td>PCIe Command – refer to section 7</td></tr>
<tr><td>5h – Fh</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>2:1</td><td>Reserved</td></tr>
<tr><td>0</td><td>Command Slot Identifier (CSI): This field indicates the Command Slot with which the message is associated. For Request Messages this field indicates the Command Slot with which the Request Message is associated. For Response Messages, this field indicates the Command Slot associated with the Request Message with which the Response Message is associated.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>Command Slot 0</td></tr>
<tr><td>1h</td><td>Command Slot 1</td></tr>
</table></td></tr>
</table>
</td></tr>
<tr><td>~~3:~~2</td><td>~~Reserved~~
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:1</td><td>Reserved</td></tr>
<tr><td>0</td><td>Management Endpoint Buffer (MEB): This field indicates whether the Request/Response Data is contained in the Request/Response Data field of this NVMe-MI Message or in the Management Endpoint Buffer. Refer to Section 3.2.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>The Request/Response Data is contained in the Request/Response Data of this NVMe-MI Message.</td></tr>
<tr><td>1h</td><td>The Request/Response Data is contained in the Management Endpoint Buffer.</td></tr>
</table></td></tr>
</table>
</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
<tr><td>N-1:4</td><td>Message Data (DATA): This field contains the NVMe-MI ~~message~~ Message payload. The format of this field depends on the NVMe-MI Message Type.</td></tr>
<tr><td>N+3:N</td><td>Message Integrity Check (MIC): This field contains a CRC computed over the contents of the message. Refer to section 3.2.1.1.</td></tr>
</table>

Modify Figure 16 in Section 4.2 as shown below:

Figure 17: Response Message Status

Value	Description	Error Reponse Format
00h	**Success:** The command completed successfully.	Refer to 4.2.1
01h	**More Processing Required:** The command is in progress and requires more time to complete processing. When this Response Message Status value is used in a Response Message, a subsequent message contains the result of the Command Message. This Response Message Status shall not be sent more than once per Request Message.	Refer to 4.2.1
02h	**Internal Error:** The command could not be executed due to a vendor specific internal error.	Refer to 4.2.1
03h	**Invalid Command Opcode:** The associated command opcode field is not valid. Invalid opcodes include reserved and optional opcodes that are not implemented.	Refer to 4.2.1
04h	**Invalid Parameter:** Invalid command parameter field value. Request Messages received with reserved values in defined fields shall be completed with an Invalid Parameter Error Response Message. Request Messages received with reserved or unimplemented values in defined fields shall be completed with an Invalid Parameter Error Response Message. Other error conditions that result in Invalid Parameter Error Response Message are noted elsewhere in this specification.	Refer to 4.2.2
05h	**Invalid Command Size:** The Command Message body was larger or smaller than that expected by the command due to a reason other than too much or too little input data (e.g., the command did not contain all the required parameters or no input data was expected but the command message body is larger than that needed to contain the required parameters). The expected command message body size is determined by the command opcode assuming no other errors are detected (e.g., Invalid Command Opcode or Invalid Parameter).	Refer to 4.2.1
06h	**Invalid Command Input Data Size:** The Command Message requires input data and contains too much or too little input data.	Refer to 4.2.1
07h	**Access Denied:** A command was prohibited from being executed due to a vendor specific protection mechanism.	Refer to 4.2.1
08h – 1Fh	Reserved	
20h	**VPD Updates Exceeded:** More updates to the VPD are attempted than allowed.	Refer to 4.2.1

21h	**PCIe Inaccessible:** The PCIe functionality is not available at this time.	Refer to 4.2.1
22h	**Management Endpoint Buffer Cleared Due to Sanitize:** An attempt was made access data in the Management Endpoint Buffer that was zeroed due to a sanitize operation.	Refer to 4.2.1
23h	**Enclosure Services Failure:** The Enclosure Services Process has failed in an unknown manner.	Refer to 4.2.1
24h	**Enclosure Services Transfer Failure:** Communication with the Enclosure Services Process has failed.	Refer to 4.2.1
25h	**Enclosure Failure:** An unrecoverable enclosure failure has been detected by the Enclosure Services Process.	Refer to 4.2.1
26h	**Enclosure Services Transfer Refused:** The NVM Subsystem or Enclosure Services Process indicated an error or an invalid format in communication.	Refer to 4.2.1
27h	**Unsupported Enclosure Function:** A SES Send command has been attempted to a simple Subenclosure.	Refer to 4.2.1
28h	**Enclosure Services Unavailable:** The NVM Subsystem or Enclosure Services Process has encountered an error, but may become available again.	Refer to 4.2.1
29h	**Enclosure Degraded:** A noncritical failure has been detected by the Enclosure Services Process.	Refer to 4.2.1
2~~2~~Ah – DFh	Reserved	

Add new Section 4.6 as shown below after Section 4.5:

4.6 Management Endpoint Buffer

Since the maximum size of the NVMe-MI Message is 4224 bytes, the maximum possible amount of out-of-band Request Data that may be contained in a Request Message is 4216 bytes (i.e., 4224 bytes minus 4 byte message header and 4 byte Message Integrity Check field) and the maximum possible amount of out-of-band Response Data that may be contained in a Response Message is 4215 bytes (i.e., 4224 bytes minus 4 byte Message Header, 1 byte Status field, and 4 byte Message Integrity Check field). The amount of supported Request or Response Data is Command Message specific due to the presence of command specific fields. In some cases it is desirable to service Command Messages that contain more Request Data or Response Data than may be transferred in an NVMe-MI Message. For example, one may wish to issue an NVM Express Admin Command Set Get Log Page command to transfer a log page that is greater in size than that allowed in the Response Data.

A Management Endpoint may support an optional Management Endpoint Buffer that facilitates Request Data and Response Data transfers that exceed that maximum size allowed by an NVMe-MI Message. Support for the Management Endpoint Buffer and its size in bytes is indicated by the Management Endpoint Buffer Size field in the Port Information Data Structure of the port with which the Management Endpoint is associated. Management Endpoints need not all have the same Management Endpoint Buffer support. For example, a subset of Management Endpoints may support a Management Endpoint Buffer and the size of each of these Management Endpoint Buffers may be different.

If a Management Endpoint supports a Management Endpoint Buffer, then all Command Messages or a subset of Command Messages supported by the Management Endpoint may support use of the Management Endpoint Buffer. A list of commands that support the use of the Management Endpoint Buffer is contained in the Management Endpoint Buffer Command Support List data structure that is retrieved using the Read NVMe-MI Data Structure command. If a Management Endpoint supports a Management Endpoint Buffer, then the Management Endpoint shall support the Management Endpoint Buffer Read and Management Endpoint Buffer Write commands.

The contents of a Management Endpoint Buffer may be read or written by a Management Controller by issuing Management Endpoint Buffer Read and Management Endpoint Buffer Write commands. The Management Endpoint Buffer is permitted to be read or written in an arbitrary manner. For example, the contents of the Management Endpoint Buffer may be written sequentially using a sequence of Management Endpoint Buffer Write commands or the contents of the Management Endpoint Buffer may be written in any order with gaps using these commands. Furthermore, Management Endpoint Buffer Read and Write commands may be interleaved allowing a portion of the Management Endpoint Buffer to be read while another portion of the Management Endpoint Buffer is written.

If the Management Endpoint Buffer (MEB) field is set to '1' in a Command Message that normally contains Request Data, then no Request Data is transferred in the Command Message itself and the required Request Data is instead transferred from the Management Endpoint Buffer. The Request Data starts at a zero offset from the start of the Management Endpoint Buffer. If the MEB field is set to '1' in a Command Message that normally contains Request Data, then the Command Message shall contain no Request Data. If the Command Message contains Request Data or is one that does not support Request Data, then the Management Endpoint responds with an Invalid Parameter error status response. The parameter with the error in this case is the Request Data field.

If the Management Endpoint Buffer (MEB) field is set to '1' in a Command Message that normally results in Response Data, then no Response Data is transferred in the corresponding Response Message itself and the Response Data is instead transferred to the Management Endpoint Buffer. The Response Data starts at a zero offset from the start of the Management Endpoint Buffer.

The contents of the Management Endpoint Buffer are set to '0' when the corresponding Management Endpoint is reset. The contents of the Management Endpoint Buffer are modified by the Management Endpoint Buffer Write command and by Command Messages that generate Response Data and have the MEB field set to '1'. When the Management Endpoint Buffer is updated with Response Data, the contents of the Management Endpoint Buffer that are not updated are set to zero (i.e., the Request/Response Data from previous Command Messages is not preserved). The same contents of the Management Endpoint Buffer may be used as Request Data for multiple Command Messages. Similarly, the Management Endpoint Buffer allows the use of Response Data generated by one Command Message to be used as the Request Data for a subsequent Command Message.

Since it is possible to have two out-of-band Command Messages, one associated with each of the two Command Slots, being simultaneously serviced that use the Management Endpoint Buffer, the Management Controller must comprehend and manage any possible race conditions. Updates to the Management Endpoint Buffer are not guaranteed to be atomic. Therefore, when a race condition involving two operations that update the Management Endpoint Buffer occurs, the final contents of the Management Endpoint Buffer may be an arbitrary mixture of the updates.

The Management Endpoint Buffer is considered a cache in the context of sanitize operations performed in an NVM Subsystem. The MCTP Management Endpoint Buffer may contain Response Data associated with a previously executed command that is not allowed during a sanitize operation. When a sanitize operation is initiated, the contents of the Management Endpoint Buffer shall be set to 0h. An attempt to access this zeroed data by a Management Endpoint Buffer Read command or any Command Message that uses the Management Endpoint Buffer, then the Management Endpoint responds with a Management Endpoint Buffer Cleared Due to Sanitize error response status. This error response is commonly associated with a Management Endpoint Buffer Read command, but may be associated with any command that uses the Management Endpoint Buffer as Request Data.

Modify Section 5 as shown below:

Figure 33: Management Interface Command Request Message Format

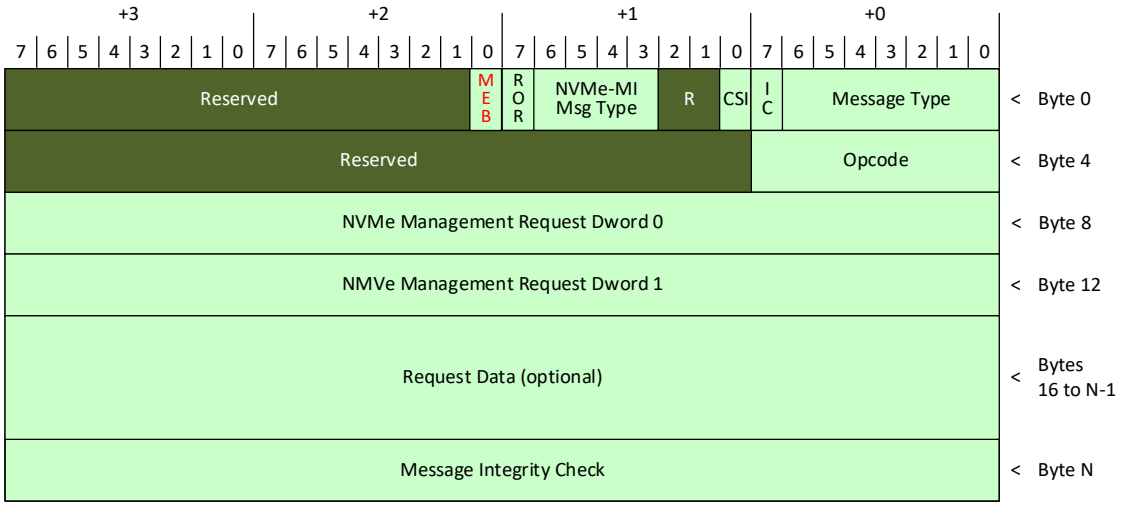


Figure 35 defines the Management Interface Command Set opcodes.

Figure 35: Opcodes for Management Interface Commands

Opcode	Command
00h	Read NVMe-MI Data Structure
01h	NVM Subsystem Health Status Poll
02h	Controller Health Status Poll
03h	Configuration Set
04h	Configuration Get
05h	VPD Read
06h	VPD Write
07h	Reset
08h	SES Receive
09h	SES Send
0Ah	Management Endpoint Buffer Read
0Bh	Management Endpoint Buffer Write
0~~8~~Ch – BFh	Reserved
C0h – FFh	Vendor specific

Figure 35a shows the Management Interface Command Set commands that are mandatory, optional, and prohibited for an NVMe Storage Device as well as for an NVMe Enclosure using the out-of-band mechanism. Figure 35b shows Management Interface Command Set commands that are mandatory, optional, and prohibited for an NVMe Storage Device as well as for an NVMe Enclosure using the in-band tunneling mechanism.

Figure 35a: Management Interface Command Support using an Out-of-Band Mechanism

NVMe Storage Device O/M/P[1]	NVMe Enclosure O/M/P[1]	Command
M	M	Read NVMe-MI Data Structure
M	O[3]	NVM Subsystem Health Status Poll
M	O[3]	Controller Health Status Poll
M	M[2]	Configuration Set
M	M[2]	Configuration Get
M	O[3]	VPD Read
M	O[3]	VPD Write
M	O[3]	Reset
P	M	SES Receive
P	M	SES Send
O	M	Management Endpoint Buffer Read
O	M	Management Endpoint Buffer Write
-	-	Reserved
O	O	Vendor specific

NOTES:
1. O/M definition: O = Optional, M = Mandatory, P = Prohibited from being supported. An NVMe Enclosure that is also an NVMe Storage Device (i.e., implements namespaces) shall implement mandatory commands required by either an NVMe Storage Device and an NVMe Enclosure and may implement optional commands allowed by either an NVMe Storage Device and an NVMe Enclosure.
2. This command was architected for an NVMe Storage Device. The mapping of Health Status Change Configuration Identifier to an NVMe Enclosure is outside the scope of this specification.
3. This command was architected for an NVMe Storage Device. The mapping of this command to an NVMe Enclosure is outside the scope of this specification.

Figure 35b: Management Interface Command Support using In-Band Tunneling Mechanism

NVMe Storage Device O/M/P[1]	NVMe Enclosure O/M/P[1]	Command
M	O[2]	Read NVMe-MI Data Structure
M	O[2]	NVM Subsystem Health Status Poll
M	O[2]	Controller Health Status Poll
M	O[2]	Configuration Set
M	O[2]	Configuration Get
M	O[2]	VPD Read
M	O[2]	VPD Write
M	O[2]	Reset
P	M	SES Receive
P	M	SES Send
P	P	Management Endpoint Buffer Read
P	P	Management Endpoint Buffer Write
-	-	Reserved
O	O	Vendor specific

NVMe Storage Device O/M/P[1]	NVMe Enclosure O/M/P[1]	Command
NOTES: 1. O/M definition: O = Optional, M = Mandatory, P = Prohibited from being supported. An NVMe Enclosure that is also an NVMe Storage Device (i.e., implements namespaces) shall implement mandatory commands required by either an NVMe Storage Device and an NVMe Enclosure and may implement optional commands allowed by either an NVMe Storage Device and an NVMe Enclosure. 2. This command was architected for an NVMe Storage Device. The mapping of this command to an NVMe Enclosure is outside the scope of this specification.		

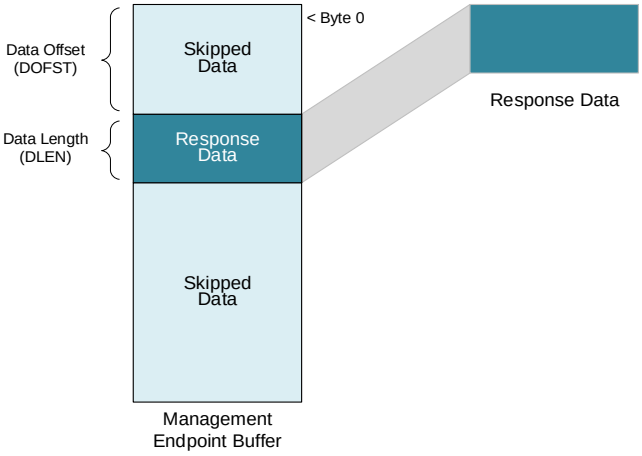
Add the following new sections after Section 5.3 as shown below and renumber remaining sections:

5.TBD Management Endpoint Buffer Read

The Management Endpoint Buffer Read command allows the Management Controller to read the contents of the Management Endpoint Buffer. This data is returned in the Response Data.

The command uses NVMe Management Dwords 0 and 1. The format of NVMe Management Dwords 0 and 1 are shown in Figure M and Figure M+1 respectively. There is no Request Data included in a Management Endpoint Buffer Read command. The NVMe Management Response field is reserved.

Figure TBD: Management Endpoint Buffer Read Response Data



If the Data Offset (DOFST) field is greater than or equal to the size of the Management Endpoint Buffer, then the Management Endpoint responds with an Invalid Parameter error status response. The parameter with the error in this case is the DOFST field. If the DOFST field is less than the size of the Management Endpoint Buffer and the sum of the DOFST and DLEN fields is greater than or equal to size of the

Management Endpoint Buffer, then the Management Endpoint responds with an Invalid Parameter error status response. The parameter with the error in this case is the DLEN field.

When an attempt is made to read Management Endpoint Buffer contents that were zeroed due to a sanitize operation, then the Management Endpoint responds with a Management Endpoint Buffer Cleared Due to Sanitize error status response .

Figure M: Management Endpoint Buffer Read – NVMe Management Dword 0

Bit	Description
31:00	**Data Offset (DOFST):** This field specifies the starting offset, in bytes, into the Management Endpoint Buffer.

Figure M+1: Management Endpoint Buffer Read – NVMe Management Dword 1

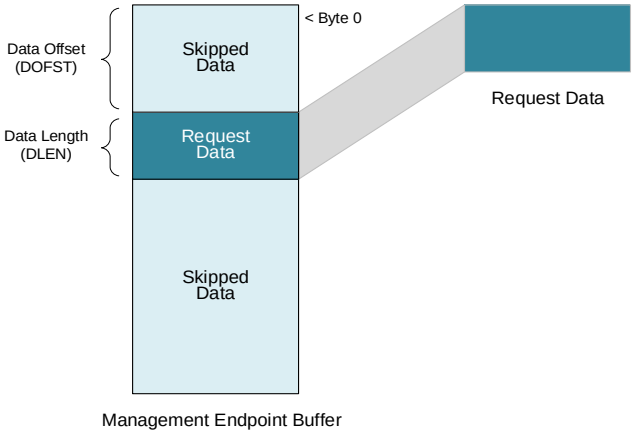
Bit	Description
31:16	Reserved
15:00	**Data Length (DLEN):** This field specifies the length, in bytes, to be transferred from the Management Endpoint Buffer starting at the byte offset specified by DOFST and returned in the Response Data. Specifying a DLEN field value that is greater than the maximum supported Response Data size results in an Invalid Parameter error status response Data Length of 0 and no data is valid. The Management Endpoint responds with a Success Response Message and no Response Data.

5.TBD Management Endpoint Buffer Write

The Management Endpoint Buffer Write command allows the Management Controller to update the contents of the optional Management Endpoint Buffer. The data used to update the Management Endpoint Buffer is transferred in the Request Data included in a Management Endpoint Buffer Write command.

The command uses NVMe Management Dwords 0 and 1. The format of the NVMe Management Dwords 0 and 1 are shown in Figure M and Figure M+1 respectively. The NVMe Management Response field is reserve and there is no Response Data.

Figure TBD: Management Endpoint Buffer Write Request Data



If the Data Offset (DOFST) field is greater than or equal to the size of the Management Endpoint Buffer, then the Management Endpoint responds with an Invalid Parameter error status response. The parameter with the error in this case is the DOFST field. If the DOFST field is less than the size of the Management Endpoint Buffer and the sum of the DOFST and DLEN fields is greater than or equal to size of the Management Endpoint Buffer, then the Management Endpoint responds with an Invalid Parameter error status response. The parameter with the error in this case is the DLEN field.

Figure M: Management Endpoint Buffer Write – NVMe Management Dword 0

Bit	Description
31:00	**Data Offset (DOFST):** This field specifies the starting offset, in bytes, into the Management Endpoint Buffer.

Figure M+1: Management Endpoint Buffer Write – NVMe Management Dword 1

Bit	Description
31:16	Reserved
15:00	**Data Length (DLEN):** This field specifies the length, in bytes, to be transferred from the Request Data to the Management Endpoint Buffer starting at the byte offset specified by DOFST. Specifying a DLEN field value that is greater than the maximum supported Response Data size results in an Invalid Parameter error status response. A DLEN value of '0' specifies that no data shall be transferred. This condition shall not be considered an error.

Modify Section 5.5 as shown below:

5.5 Read NVMe-MI Data Structure

The Read NVMe-MI Data Structure command requests data that describes information about the NVM Subsystem, the Management Endpoint or the NVMe Controllers.
The command uses NVMe Management Dword 0. The format of NVMe Management Dword 0 is shown in Figure 60. NVMe Management Dword 1 is reserved. There is no Request Data included in a Read NVMe-MI Data Structure command.

Figure 60: Read NVMe-MI Data Structure – NVMe Management Dword 0

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>31:24</td><td>Data Structure Type (DTYP): This field specifies the data structure to return

<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00h</td><td>NVM Subsystem Information</td></tr>
<tr><td>01h</td><td>Port Information</td></tr>
<tr><td>02h</td><td>Controller List</td></tr>
<tr><td>03h</td><td>Controller Information</td></tr>
<tr><td>04h</td><td>Optional Commands Supported</td></tr>
<tr><td>05h</td><td>Management Endpoint Buffer Command Support List</td></tr>
<tr><td>0<s>5</s>6h-FFh</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>23:16</td><td>Port Identifier (PORTID): This field contains the identifier of the port whose data structure is returned.

If the DTYP field value corresponds to Port Information, then this field contains the Port Identifier whose information is requested.

If the DTYP field value corresponds to Management Endpoint Buffer Command Support List, then this field contains the Port Identifier whose information is requested.

For all other values of the DTYP field, this field is reserved.</td></tr>
<tr><td>15:00</td><td>Controller Identifier (CTRLID): This field contains the Controller identifier whose data structure is returned.

If the DTYP field value corresponds to Controller List or Controller Information, then this field contains the Controller identifier in the NVM Subsystem whose information is requested.

For all other values of the DTYP field, this field is reserved.</td></tr>
</table>

Upon successful completion of the Read NVMe-MI Data Structure, the NVMe Management Response field is shown in Figure 61 and the specified data structure is returned in the Response Data.

Figure 61: Read NVMe-MI Data Structure – NVMe Management Response

Bit	Description
23:16	Reserved
15:00	**Response Data Length:** The length, in bytes, of the Response Data field in this Response Message.

The NVM Subsystem Information data structure contains information about the NVM Subsystem. The Port Identifier and Controller Identifier fields are reserved. The format is shown in Figure 62.

Figure 62: NVM Subsystem Information Data Structure

Byte	Description
00	**Number of Ports (NUMP):** This field specifies the maximum number of ports of any type supported by the NVM Subsystem. This is a 0's based value.
01	**NVMe-MI Major Version Number (MJR):** This field shall be set to 1h to indicate the major version number of this specification.
02	**NVMe-MI Minor Version Number (MNR):** This field shall be cleared to 0h to indicate the minor version number of this specification.
31:03	Reserved

The Port Information data structure contains information about a port within the NVM Subsystem. The Port Identifier specifies the port. The Controller Identifier fields are reserved. The format is shown in Figure 63.

Figure 63: Port Information Data Structure

Byte	Description
00	**Port Type:** Specifies the port type. Value / Definition: 0h – Inactive 1h – PCIe 2h – SMBus 3h – FFh – Reserved
01	Reserved
03:02	**Maximum MCTP Transmission Unit Size:** The maximum MCTP Transmission Unit size the port is capable of sending and receiving. If the port does not support MCTP, then this field shall be set to 0. If the port type is PCIe and the port supports MCTP, then this field shall be set to a value between 64 bytes and the PCIe Max Payload Size supported minus 4, inclusive. All PCIe ports within an NVM Subsystem should report the same value in this field. If the port type is SMBus and the port supports MCTP, then this field shall be set to a value between 64 bytes and 250 bytes, inclusive.
07:04	~~Reserved~~ **Management Endpoint Buffer Size**: This field specifies the size of the Management Endpoint Buffer in bytes when a Management Endpoint Buffer is supported. A value of 0000h in this field indicates that the Management Endpoint does not support a Management Endpoint Buffer.
31:08	Port Type Specific (refer to Figure 64 and Figure 65)

Figure 64: PCIe Port Specific Data

<table>
<tr><th>Byte</th><th>Description</th></tr>
<tr><td>08</td><td>PCIe Maximum Payload Size: This field indicates the Max Payload Size for the specified PCIe port. If the link is not active, this field should be cleared to 0h.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0h</td><td>128 bytes</td></tr>
<tr><td>1h</td><td>256 bytes</td></tr>
<tr><td>2h</td><td>512 bytes</td></tr>
<tr><td>3h</td><td>1024 bytes</td></tr>
<tr><td>4h</td><td>2048 bytes</td></tr>
<tr><td>5h</td><td>4096 bytes</td></tr>
<tr><td>6h-FFh</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>09</td><td>PCIe Supported Link Speeds Vector: This field indicates the Supported Link Speeds for the specified PCIe port.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7:3</td><td>Reserved</td></tr>
<tr><td>2</td><td>This bit shall be set to ‘1’ if the link supports 8.0 GT/s</td></tr>
<tr><td>1</td><td>This bit shall be set to ‘1’ if the link supports 5.0 GT/s</td></tr>
<tr><td>0</td><td>This bit shall be set to ‘1’ if the link supports 2.5 GT/s.</td></tr>
</table></td></tr>
<tr><td>10</td><td>PCIe Current Link Speed: The port’s PCIe negotiated link speed using the same encoding as the PCIe Supported Link Speed Vector field. A value of 0h in this field indicates the PCIe Link is not available.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0h</td><td>Link not active</td></tr>
<tr><td>1h</td><td>The current link speed is the speed indicated in the supported link speed bit 0.</td></tr>
<tr><td>2h</td><td>The current link speed is the speed indicated in the supported link speed bit 1.</td></tr>
<tr><td>3h</td><td>The current link speed is the speed indicated in the supported link speed bit 2.</td></tr>
<tr><td>4h</td><td>The current link speed is the speed indicated in the supported link speed bit 3.</td></tr>
<tr><td>5h</td><td>The current link speed is the speed indicated in the supported link speed bit 4.</td></tr>
<tr><td>6h</td><td>The current link speed is the speed indicated in the supported link speed bit 5.</td></tr>
<tr><td>7h</td><td>The current link speed is the speed indicated in the supported link speed bit 6.</td></tr>
<tr><td>8h-FFh</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>11</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.
<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>5-7</td><td>Reserved</td></tr>
<tr><td>8</td><td>PCIe x8</td></tr>
<tr><td>9-11</td><td>Reserved</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table>
<tr><td>12</td><td>PCIe x12</td></tr>
<tr><td>13-15</td><td>Reserved</td></tr>
<tr><td>16</td><td>PCIe x16</td></tr>
<tr><td>17-31</td><td>Reserved</td></tr>
<tr><td>32</td><td>PCIe x32</td></tr>
<tr><td>33-255</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>12</td><td>PCIe Negotiated Link Width: The negotiated PCIe link width for this port.<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0</td><td>Link not active</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>5-7</td><td>Reserved</td></tr>
<tr><td>8</td><td>PCIe x8</td></tr>
<tr><td>9-11</td><td>Reserved</td></tr>
<tr><td>12</td><td>PCIe x12</td></tr>
<tr><td>13-15</td><td>Reserved</td></tr>
<tr><td>16</td><td>PCIe x16</td></tr>
<tr><td>17-31</td><td>Reserved</td></tr>
<tr><td>32</td><td>PCIe x32</td></tr>
<tr><td>33-255</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>31:13</td><td>Reserved</td></tr>
</table>

Figure 65: SMBus Port Specific Data

Byte	Description
08	**Current VPD SMBus/I2C Address:** This field indicates the current VPD SMBus/I2C address. A value of 0h indicates there is no VPD.
09	**Maximum VPD Access SMBus/I2C Frequency:** This field indicates the maximum SMBus/I2C frequency supported on the VPD interface.<table><tr><th>Value</th><th>Definition</th></tr><tr><td>0h</td><td>Not supported</td></tr><tr><td>1h</td><td>100 kHz</td></tr><tr><td>2h</td><td>400 kHz</td></tr><tr><td>3h</td><td>1 MHz</td></tr><tr><td>4-FFh</td><td>Reserved</td></tr></table>
10	**Current Management Endpoint SMBus/I2C Address:** This field indicates the current MCTP SMBus/I2C address. A value of 0h indicates there is no Management Endpoint on this port.
11	**Maximum Management Endpoint SMBus/I2C Frequency:** This field indicates the maximum SMBus/I2C frequency supported by the Management Endpoint.<table><tr><th>Value</th><th>Definition</th></tr><tr><td>0h</td><td>Not supported</td></tr><tr><td>1h</td><td>100 kHz</td></tr><tr><td>2h</td><td>400 kHz</td></tr><tr><td>3h</td><td>1 MHz</td></tr><tr><td>4-FFh</td><td>Reserved</td></tr></table>
12	**NVMe Basic Management:** Bit 0 in this field, if set to '1', indicates if the port implements the NVMe Basic Management command specified in Appendix A. All other bits in this field are reserved.
31:13	Reserved

The Controller List data structure contains a list of NVMe Controllers in the NVM Subsystem greater than or equal to the value specified in the Controller Identifier (CTRLID) field. A Controller List may contain up to 2047 Controller identifiers. Refer to the NVM Express specification for a definition of the Controller List data structure.

Figure 66: Controller Information Data Structure

<table>
<tr><th>Byte</th><th>Description</th></tr>
<tr><td>00</td><td>Port Identifier (PORTID): This field specifies the PCIe Port Identifier with which the Controller is associated.</td></tr>
<tr><td>04:01</td><td>Reserved</td></tr>
<tr><td>05</td><td>PCIe Routing ID Information (PRII): This field provides additional data about the PCI Express Routing ID (PRI) for the specified Controller.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7:1</td><td>Reserved</td></tr>
<tr><td>0</td><td>PCIe Routing ID Valid: This bit is set to ‘1’ if the device has captured a Bus Number and Device Number (Bus Number only for ARI devices). This bit is set to ‘0’ if the device has not captured a Bus and Device number (Bus Number only for ARI devices).</td></tr>
</table></td></tr>
<tr><td>07:06</td><td>PCIe Routing ID (PRI): This field contains the PCIe Routing ID for the specified Controller.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>15:8</td><td>PCI Bus Number: The Controller’s PCI Bus Number.</td></tr>
<tr><td>7:3</td><td>PCI Device Number: The Controller’s PCI Device Number.</td></tr>
<tr><td>2:0</td><td>PCI Function Number: The Controller’s PCI Function Number.</td></tr>
</table>
Note: For an ARI Device, bits 7:0 represents the (8-bit) Function Number, which replaces the (5-bit) Device Number and (3-bit) Function Number fields above.</td></tr>
<tr><td>09:08</td><td>PCI Vendor ID: The PCI Vendor ID for the specified Controller.</td></tr>
<tr><td>11:10</td><td>PCI Device ID: The PCI Device ID for the specified Controller.</td></tr>
<tr><td>13:12</td><td>PCI Subsystem Vendor ID: The PCI Subsystem Vendor ID for the specified Controller.</td></tr>
<tr><td>15:14</td><td>PCI Subsystem Device ID: The PCI Subsystem Device ID for the specified Controller.</td></tr>
<tr><td>31:16</td><td>Reserved</td></tr>
</table>

The Optionally Supported Command List data structure contains a list of optional commands that a Management Endpoint supports. The Optionally Supported Command List data structure may contain up to 2047 commands, and shall be minimally sized (i.e., if there is 1 optionally supported command, the data structure is 4 bytes total).

Figure 67 Optionally Supported Command List Data Structure

Byte	Description
01:00	**Number of Commands (NUMCMD):** This field contains the number of optionally supported commands in the list. A value of 0h indicates there are no commands in the list.
03:02	**Command 0 (CMD0):** This field contains the Command Type and Opcode for the first optionally supported command or 0h if the list is empty (i.e. no optional commands are supported). Refer to Figure 68.

05:04	**Command 1 (CMD1):** This field contains the Command Type and Opcode for the second optionally supported command, if applicable. Refer to Figure 68.
…	
(N*2 +3): (N*2 + 2)	**Command N (CMDN):** This field contains the Command Type and Opcode for the N+1 optionally supported command, if applicable. Refer to Figure 68.

Figure 68: Optionally Supported Command Data Structure

Byte	Description
00	**Command Type:** This field specifies the command set used by the optionally supported command. Bits 7: Reserved Bits 6:3: **NVMe-MI Message Type (NMIMT):** This field specifies the NVMe-MI Message Type. Refer to Figure 11. Bits 2:0: Reserved
01	**Opcode:** This field specifies the opcode used for the optionally supported command.

If the Management Endpoint Buffer Size field in the Port Information Data Structure is not 0000h, then returning of the Management Endpoint Buffer Command Support List data structure shall be supported by the Management Endpoint. If the Management Endpoint Buffer Size field in the Port Information Data structure is 0000h, then the Data Structure Type value for Management Endpoint Buffer Command Support List is reserved.

The Management Endpoint Buffer Command Support List data structure contains a list of commands that support the use of the Management Endpoint Buffer. The data structure may contain up to 2047 commands, and shall be minimally sized (i.e., if there is 1 optionally supported command, the data structure is 4 bytes total).

The list of commands that support the Management Endpoint Buffer may be different among Management Endpoints within the NVM Subsystem. The Port Identifier (PORTID) field in NVMe Management Dword 0 of the Read NVMe-MI Data Structure specifies the port of the Management Endpoint whose Management Endpoint Buffer Command Support List data structure is returned.

Figure 68a: Management Endpoint Buffer Supported Command List Data Structure

Byte	Description
01:00	**Number of Commands (NUMCMD):** This field contains the number of commands in the list. A value of 0000h indicates there are no commands in the list.
03:02	**Command 0 (CMD0):** This field contains the Management Endpoint Buffer Supported Command Data Structure (refer to Figure 68b) for the first command that supports the use of the Management Endpoint Buffer associated with the Management Endpoint.
05:04	**Command 1 (CMD1):** This field contains the Management Endpoint Buffer Supported Command Data Structure (refer to Figure 68b) for the second command that supports the use of the Management Endpoint Buffer associated with the Management Endpoint.
…	
(N*2 +3): (N*2 + 2)	**Command N (CMDN):** This field contains the Management Endpoint Buffer Supported Command Data Structure (refer to Figure 68b) for the N+1 command that supports the use of the Management Endpoint Buffer associated with the Management Endpoint.

Figure 68b: Management Endpoint Buffer Supported Command Data Structure

<table>
<tr><th>Byte</th><th>Description</th></tr>
<tr><td>00</td><td>Command Type: This field specifies the command set that supports the Management Endpoint Buffer.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7</td><td>Reserved</td></tr>
<tr><td>6:3</td><td>NVMe-MI Message Type (NMIMT): This field specifies the NVMe-MI Message Type. Refer to Figure 11.</td></tr>
<tr><td>2:0</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>01</td><td>Opcode: This field specifies the opcode of the command that supports the Management Endpoint Buffer.</td></tr>
</table>

Add the following new sections after Section 5.6 as shown below and renumber remaining sections:

5.7 SES Receive

The SES Receive command is used to retrieve SES status type diagnostic pages. Upon successful completion of the SES Receive command, the SES status type diagnostic page is returned in the Response Data.

The SES Receive command uses NVMe Management Dwords 0 and 1. The format of NVMe Management Dword 0 is shown in Figure Xa and the format of NVMe Management Dword 1 is shown in Figure Xb. There is no Request Data sent in the Request Message.

The Page Code (PCODE) field specifies the SES status type diagnostic page to be retrieved. Refer to SES-3 for a list and description of SES diagnostic pages. If the PCODE field specifies a reserved value, an unsupported value, or a value that only corresponds to a SES control type diagnostic page, then the Management Endpoint responds with an Invalid Parameter error status response.

The Allocation Length (ALENGTH) field specifies the maximum length of the Response Data field in the Response Message and is used to limit the maximum amount of SES diagnostic page data that may be returned. The length of the Response Data field shall be the total length of the SES diagnostic page specified by the PCODE field or the number of bytes specified by the ALENGTH field (i.e., the SES diagnostic page is truncated), whichever is less. When the SES diagnostic page is truncated, the value of fields within the SES diagnostic page are not altered to reflect the truncation.

All errors are detected and reported while servicing the SES Receive command and reported via an Error Response. If an invalid field is detected in a SES Receive command, then the Management Endpoint responds with an Invalid Parameter error status response. If a condition occurs that in SES-3 results in a CHECK CONDITION, then the Management Endpoint responds with an Error Response. The mapping of Error Responses to SES-3 sense keys and additional sense codes is shown in Figure F+4.

If the SES Receive command is supported in the out-of-band mechanism, then the Management Endpoint shall support the use of the Management Endpoint Buffer with SES Receive command and the size of the Management Endpoint Buffer shall be greater than or equal to the maximum supported SES status type diagnostic page. This allows a Management Controller to retrieve an SES status type diagnostic page whose size exceed the maximum size allowed by one NVMe-MI Message.

The amount of data returned in the Response Data or transferred to the Management Endpoint Buffer is dependent on the SES status diagnostic page that is returned. The Response Data Length field in the NVMe Management Response contains the length of the Response Data.

Figure Xa: SES Receive – NVMe Management Dword 0

Bit	Description
31:8	Reserved
07:00	**Page Code (PCODE):** This field specifies the SES status diagnostic page to be transferred.

Figure Xb: SES Receive – NVMe Management Dword 1

Bit	Description
31:16	Reserved
15:00	**Allocation Length (ALENGTH):** This field specifies the maximum length of the Response Data field in the Response Message.

Figure X+1: SES Receive – NVMe Management Response

Bit	Description
23:16	Reserved
15:00	**Response Data Length (RDL):** The length, in bytes, of the Response Data field in this Response Message or transferred to the Management Endpoint Buffer.

5.8 SES Send

The SES Send command is used to transfer SES control type diagnostic pages to an SES Enclosure Service Process. Upon successful completion of the SES Send command, the Request Data, containing an SES control type diagnostic page, is transferred by the Request Message or to the Management Endpoint Buffer.

Unlike the SES Receive command that specifies the page code of the SES status diagnostic page being retrieved, the SES Send command specifies the page code of the SES control type diagnostic page that is being transferred in the SES control type diagnostic page itself. Refer to SES-3 for a list and description of SES control type diagnostic pages. If the page code in the SES control type diagnostic page specifies a reserved value, an unsupported value, or a value that only corresponds to an SES status diagnostic page, then the Management Endpoint responds with an Invalid Parameter error status response.

The SES Send command does not use NVMe Management Dword 0 or the NVMe Management Response field. All of these are reserved.

All errors are detected and reported while processing the SES Send command and reported via an Error Response. If an invalid field is detected in the SES control type diagnostic page data transferred by an SES Send command, then the Management Endpoint responds with an Invalid Parameter error status response. If a condition occurs that in SES-3 results in a CHECK CONDITION, then the Management Endpoint responds with an Error Response. The mapping of Response Message Status to SES-3 sense keys and additional sense codes is shown in Figure F+4.

The length in bytes of the Request Data field is specified in the Data Length (DLEN) field in NVMe Management Dword 1. An SES Send command with DLEN equal to 0 and no data is valid, and results in a Success Response Message. If the DLEN field specifies a value that is greater than PAGE LENGTH field in the SES control type diagnostic page plus four, then the extra data in the Request Data field following the page is ignored. If the DLEN field specifies a value that is less than PAGE LENGTH field in the SES control type diagnostic page plus four, then the page is processed using the data contained in the Request Data field.

If the SES Send command is supported in the out-of-band mechanism, then the Management Endpoint shall support the use of the Management Endpoint Buffer with the SES Send command and the size of the Management Endpoint Buffer shall be greater than or equal to the maximum supported SES control type diagnostic page. This allows a Management Controller to transfer an SES control type diagnostic page whose size exceeds the maximum size allowed by one NVMe-MI Message.

Figure Xc: SES Send – NVMe Management Dword 1

Bit	Description
31:16	Reserved
15:00	**Data Length (DLEN):** This field specifies the Request Data field in bytes.

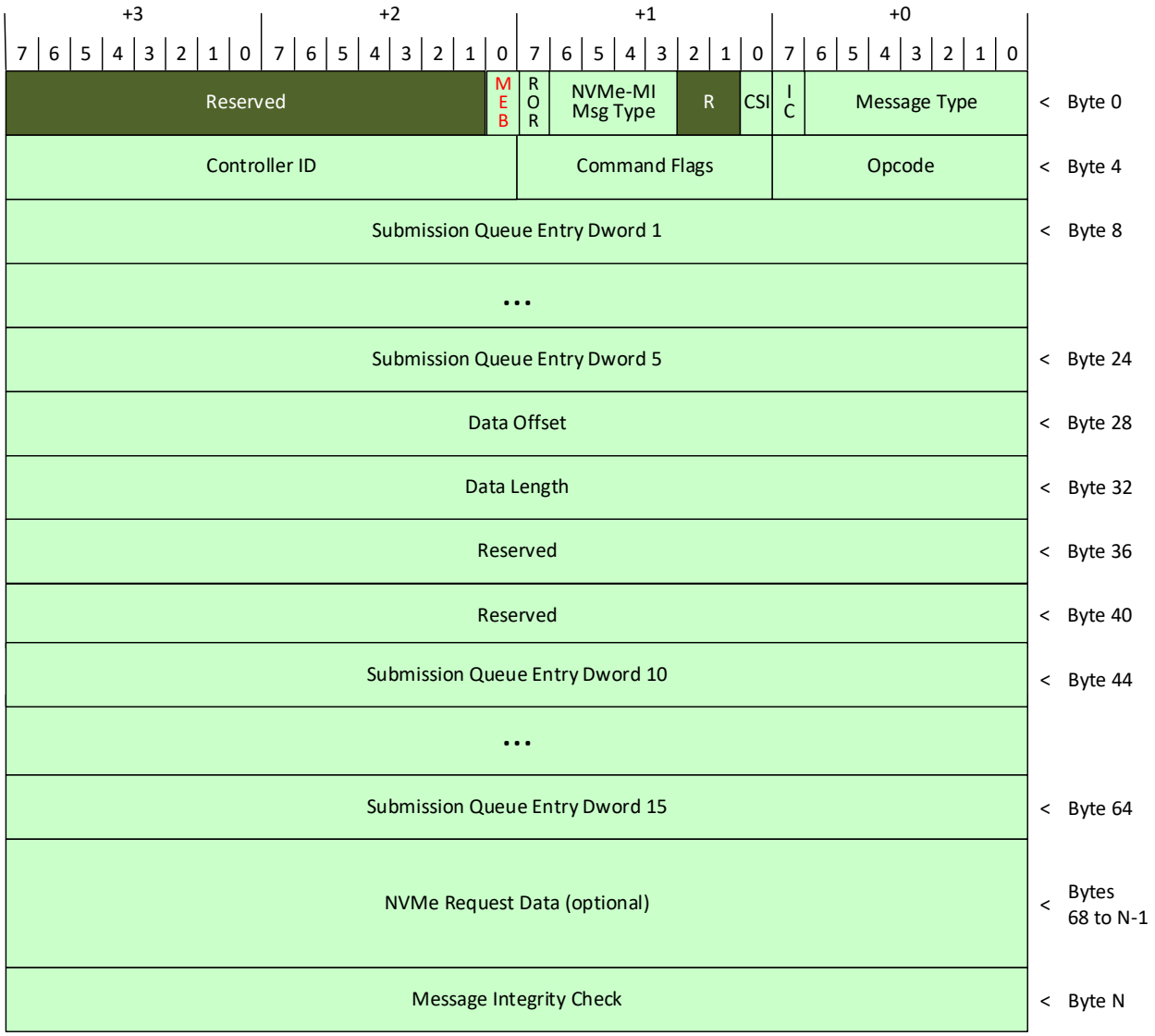
Modify Section 6 as shown below:

The NVM Express Admin Command Set allows NVMe Admin commands to be issued to any Controller in the NVM Subsystem using NVMe-MI. Figure 76 shows NVM Express Admin Commands that are mandatory, optional, and prohibited for an NVMe Storage Device and an NVMe Enclosure using the out-of-band mechanism. All NVM Express Admin Commands are prohibited using the in-band tunneling mechanism. ~~Supported commands are listed in Figure 76, and~~ The commands are defined in the NVM Express specification. If an NVMe Admin Command is issued in a Request Message other than one listed in Figure 76, the Management Endpoint shall return a response with status Invalid Parameter pointing to the NVMe opcode. Future revisions of this specification may add additional commands to Figure 76Figure .

Figure 76: List of NVMe Admin Commands Supported using the Out-of-Band Mechanism

Command	NVMe Storage Device O/M/P[1]	NVMe Enclosure O/M/P[1]
Firmware Activate/Commit	O	O
Firmware Image Download	O	O
Format NVM	O	P
Get Features	M	O
Get Log Page	M	O
Identify	M	O
Namespace Management	O	P
Namespace Attachment	O	P
Security Send	O	P
Security Receive	O	P
Set Features	O	P
Vendor Specific	O	O
NOTES: 1 O/M definition: O = Optional, M = Mandatory, P = Prohibited from being supported. An NVMe Enclosure that is also an NVMe Storage Device (i.e., implements namespaces) shall implement mandatory commands required by either an NVMe Storage Device and an NVMe Enclosure and may implement optional commands allowed by either an NVMe Storage Device and an NVMe Enclosure.		

Figure 77: NVMe Admin Command Request Format



Modify Section 7 as shown below:

Figure 84 defines the PCIe Command opcodes. It also shows PCIe Commands that are mandatory, optional, and prohibited for an NVMe Storage Device and an NVMe Enclosure using the out-of-band mechanism. All PCIe Commands are prohibited using the in-band tunneling mechanism.

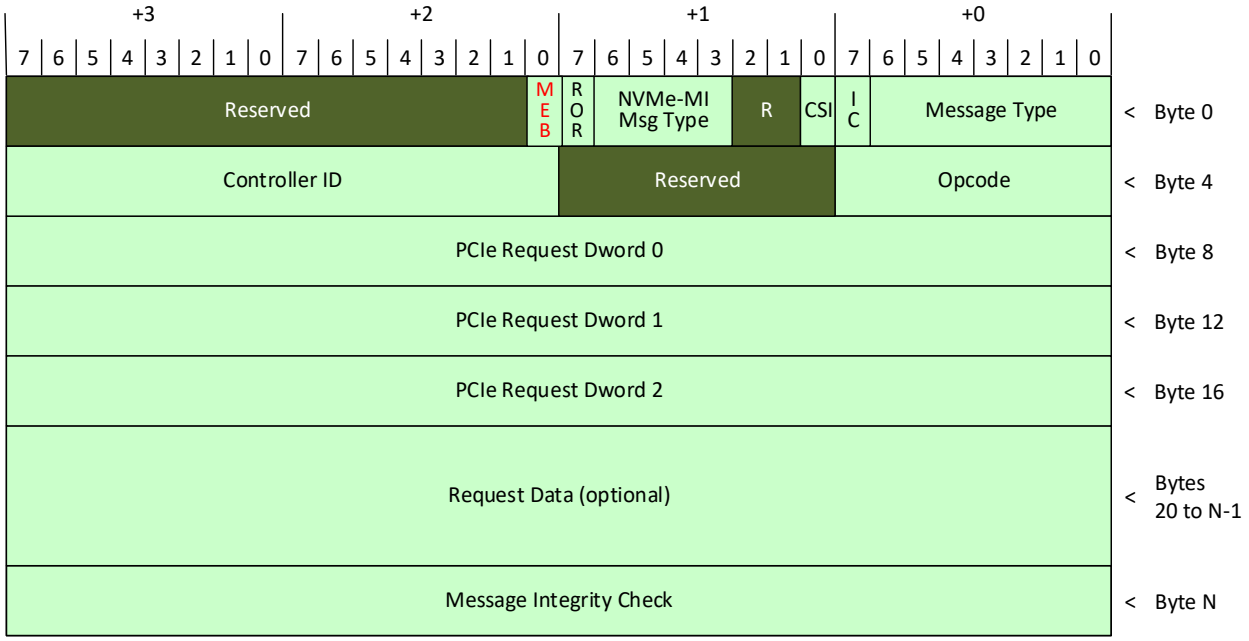
Figure 84: Opcodes for PCIe Commands using an Out-of-Band Mechanism

Opcode	NVMe Storage Device O/M/P[1]	NVMe Enclosure O/M/P[1]	Command
00h	O	O	PCIe Configuration Read
01h	O	O	PCIe Configuration Write
02h	O	O	PCIe Memory Read
03h	O	O	PCIe Memory Write
04h	O	O	PCIe I/O Read

Opcode	NVMe Storage Device O/M/P[1]	NVMe Enclosure O/M/P[1]	Command
05h	O	O	PCIe I/O Write
06h – FFh	-	-	Reserved

NOTES:
1. O/M definition: O = Optional, M = Mandatory, P = Prohibited from being supported. An NVMe Enclosure that is also an NVMe Storage Device (i.e., implements namespaces) shall implement mandatory commands required by either an NVMe Storage Device and an NVMe Enclosure and may implement optional commands allowed by either an NVMe Storage Device and an NVMe Enclosure.

Figure 82: PCIe Command Request Format



Modify Section 8.1 as shown below:

8.1 Identify Controller

The NVMe Identify Controller data structure contains information about an NVMe Controller. Bytes 240-255 have been allocated by the NVM Express specification for NVMe-MI are defined below.

Figure 101: NVMe Management Interface Identify Controller

Bytes	O/M	Description
~~254~~252:240		Reserved

<table>
<tr><td>253</td><td>M</td><td>NVM Subsystem Report (NVMSR): This field reports information associated with the NVM subsystem. At least one bit in this field shall be set to ‘1’.

<table><tr><th>Bit</th><th>Definition</th></tr><tr><td>0</td><td>NVMe Storage Device (NVMESD): If set to ‘1’, then the NVM Subsystem is part of an NVMe Storage Device. If cleared to ‘0’, then the NVM subsystem is not part of an NVMe Storage Device.</td></tr><tr><td>1</td><td>NVMe Enclosure (NVMEE): If set to ‘1’, then the NVM Subsystem is part of an NVMe Enclosure. If cleared to ‘0’, then the NVM subsystem is not part of an NVMe Enclosure.</td></tr><tr><td>7:2</td><td>Reserved</td></tr></table></td></tr>
<tr><td>254</td><td></td><td>Reserved</td></tr>
<tr><td>255</td><td>M</td><td>Management Endpoint Capabilities (MEC): This field indicates the capabilities of the Management Endpoint in the Controller.
~~Bits 7:2 are reserved.~~
~~Bit 1: If set to ‘1’ then the NVM Subsystem contains a Management Endpoint on a PCIe port.~~
~~Bit 0: If set to ‘1’ then the NVM Subsystem contains a Management Endpoint on an SMBus/I2C port.~~

<table><tr><th>Bit</th><th>Definition</th></tr><tr><td>0</td><td>SMBus/I2C Port Management Endpoint (SMBUSME): If set to ‘1’ then the NVM Subsystem contains a Management Endpoint on an SMBus/I2C port.</td></tr><tr><td>1</td><td>PCIe Port Management Endpoint (PCIEME): If set to ‘1’ then the NVM Subsystem contains a Management Endpoint on a PCIe port.</td></tr><tr><td>7:2</td><td>Reserved</td></tr></table></td></tr>
</table>