NVM Express®

Management Interface

Revision 1.2a

July 22nd, 2021

Please send comments to info@nvmexpress.org

NVM Express® Management Interface revision 1.2a specification available for download at http://nvmexpress.org. NVM Express Management Interface revision 1.2a specification consists of the NVM Express Management Interface revision 1.2 specification (refer to https://nvmexpress.org/changes-in-nvme-mi-revision-1-2 for details), ECN 001, and the implementation note in the UDID Type field description that describes which versions that note applies to.

SPECIFICATION DISLAIMER

LEGAL NOTICE:

This NVM Express® Management Interface Revision 1.2a 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 1.2a 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: **"© 2007-2021 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.

The NVM Express® design mark is a registered trademark of NVM Express, Inc.
PCI-SIG®, PCI Express®, and PCIe® are registered trademarks of PCI-SIG.

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

Table of Contents

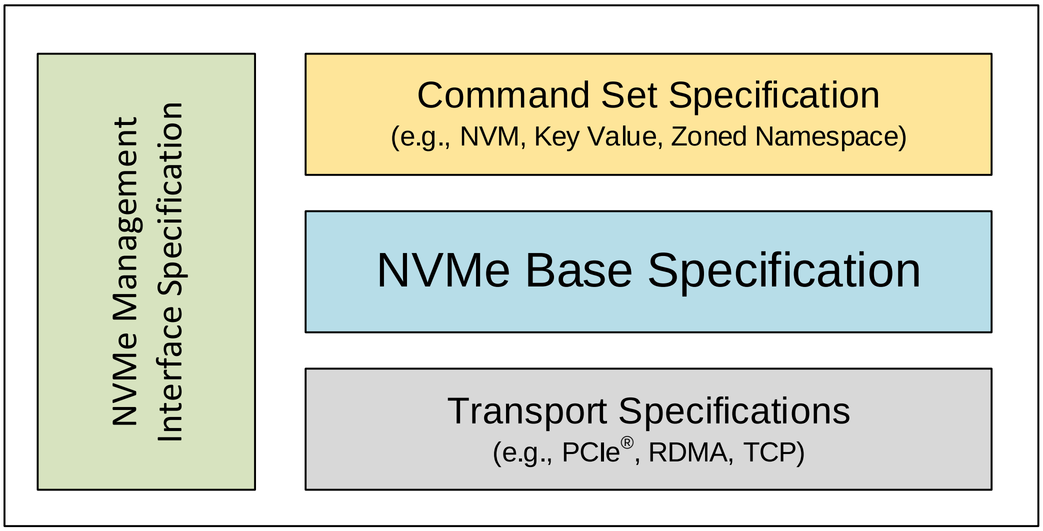
Table of Figures

1 Introduction

1.1 Overview

The NVM Express® Management Interface Specification is a member of the NVMe Family of Specifications displayed in Figure 1.

Figure 1: NVMe Family of Specifications


NVMe Management Interface Specification
Command Set Specification
(e.g., NVM, Key Value, Zoned Namespace)
NVMe Base Specification
Transport Specifications
(e.g., PCIe®, RDMA, TCP)

The NVM Express® (NVMe®) interface allows in-band host software to communicate with an NVM Subsystem. Since this specification builds on the NVM Express® Base Specification, knowledge of the NVM Express® Base Specification is assumed.

This specification defines several mechanisms to manage NVMe Storage Devices (refer to section 1.8.18) or NVMe Enclosures (refer to section 1.8.16). One mechanism allows a Management Controller to communicate out-of-band with an NVMe Storage Device or NVMe Enclosure over one or more external interfaces. Another mechanism is the in-band tunneling mechanism which allows the NVMe-MI Management Interface Command Set to be tunneled in-band via the NVMe Admin Commands NVMe-MI Send and NVMe-MI Receive to an NVMe Storage Device or NVMe Enclosure. Refer to the NVM Express Base Specification and section 4.3 of this specification for additional details on the NVMe-MI Send and NVMe-MI Receive commands.

NVMe Storage Devices and NVMe Enclosures that comply with this specification are allowed to support only the out-of-band mechanism, only the in-band tunneling mechanism, or both the out-of-band mechanism and in-band tunneling mechanism.

1.2 Scope

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

This specification defines the following key aspects for NVMe Storage Devices:

- Discover NVMe Storage Devices that are present and learn capabilities of each NVMe Storage Device;

- Store data about the host environment enabling a Management Controller or other entity to query the data later;
- Health and temperature monitoring;
- Multiple concurrent commands to prevent a long latency command from blocking monitoring operations;
- An out-of-band mechanism that is host processor and operating system agnostic;
- A standard format for VPD and defined mechanisms to read/write VPD contents; and
- Preserves data-at-rest security.

This specification defines the following key aspects for NVMe Enclosures:

- Discover NVMe Enclosures and learn their capabilities;
- Manage and sense the state of NVMe Enclosure elements such as power supplies, cooling devices, displays, and indicators;
- Multiple concurrent commands to prevent a long latency command from blocking monitoring operations;
- An out-of-band mechanism that is host processor and operating system agnostic;
- Discover NVMe Storage Devices that are present in Enclosure slots; and
- 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 the NVMe interface 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 of this specification.

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 over Fabrics 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 PCIe® switches. Coordination between multiple Requesters or a Requester and a device other than a Responder is outside the scope of this specification. Refer to section 1.8 for the definitions of Requester and Responder.

Coordinating concurrency resulting from operations associated with multiple Responders or between 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 NVMe Enclosure may contain multiple Enclosure Services Processes. Communication and coordination between the Enclosure Services Processes that manage NVMe Enclosure elements is outside the scope of this specification.

1.3 Theory of Operation

This specification is designed to provide a common interface over multiple physical layers (i.e., PCI Express, SMBus/I2C) for inventory, monitoring, configuration, and change management. This specification provides the flexibility necessary to manage NVMe Storage Devices or NVMe Enclosures using an out-of-band mechanism or in-band tunneling mechanism in a variety of host environments and systems. This specification also defines a FRU Information Device that contains Vital Product Data (refer to section 1.3.1.2).

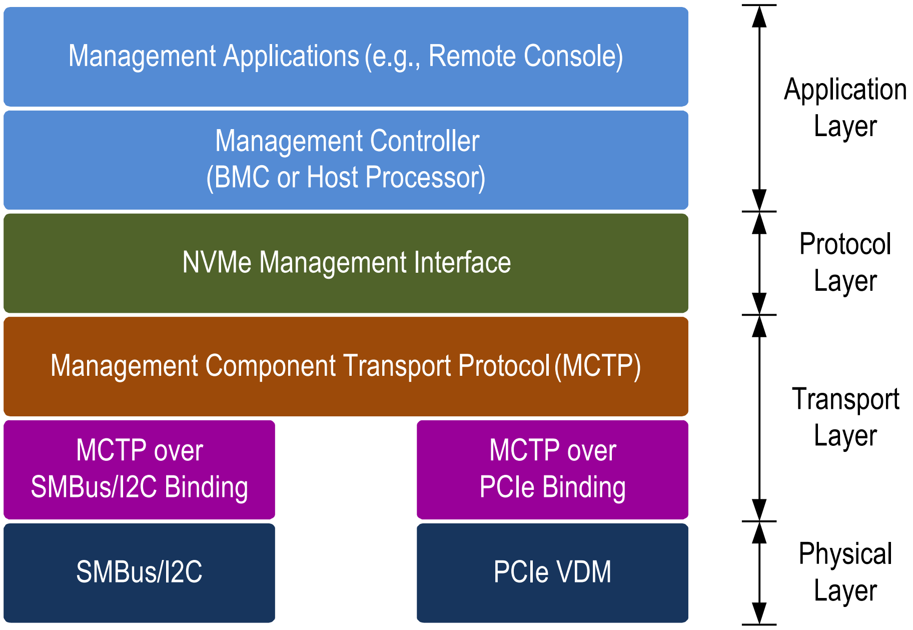
1.3.1 Out-of-Band Theory of Operation

This specification defines a mechanism for managing NVMe Storage Devices and NVMe Enclosures out-of-band via the Management Component Transport Protocol.

1.3.1.1 Management Component Transport Protocol

The out-of-band mechanism utilizes the Management Component Transport Protocol (MCTP) as the transport and utilizes existing MCTP SMBus/I2C and PCIe bindings for the physical layer. Command Messages are submitted to one of two Command Slots associated with a Management Endpoint contained in an NVM Subsystem. Figure 2 shows the NVMe-MI out-of-band protocol layering.

Figure 2: NVMe-MI Out-of-Band Protocol Layering



1.3.1.2 FRU Information Device

This specification defines a mechanism to access a FRU Information Device either via SMBus/I2C as defined by the IPMI Platform Management FRU Information Storage Definition specification or via the VPD Read and VPD Write commands. The data stored in the FRU Information Device is referred to as Vital Product Data (refer to section 8.2). A FRU Information Device may be implemented in a variety of ways (e.g., a serial EEPROM, one-time programmable memory in an NVMe Controller ASIC, etc.).

1.3.2 In-Band Theory of Operation

This specification defines an in-band tunneling mechanism that utilizes the NVMe Admin Commands NVMe-MI Send and NVMe-MI Receive. Refer to the NVM Express Base Specification and section 4.3 of this specification for additional details on the NVMe-MI Send and NVMe-MI Receive commands.

1.4 NVM Subsystem Architectural Model

This specification defines an interface that may be used to manage NVM Subsystems contained within an NVMe Storage Device or NVMe Enclosure.

The NVMe Storage Device (NVMESD) bit in the NVM Subsystem Report (NVMSR) field of the Identify Controller data structure shall be set to ‘1’ for an NVMe Storage Device. The NVMe Enclosure (NVMEE) bit in the NVM Subsystem Report (NVMSR) field of the Identify Controller data structure shall be set to ‘1’ for an NVMe Enclosure. The NVMESD bit, the NVMEE bit, or both the NVMESD bit and the NVMEE bit shall be set to ‘1’ (refer to the NVM Express Base Specification).

Management of an NVM Subsystem using the in-band tunneling mechanism and the out-of-band mechanism consists of sending Command Messages and receiving corresponding Response Messages. Command Messages consist of:

a) standard NVMe Admin Commands that target a Controller within the NVM Subsystem;
b) commands that provide access to the PCI Express configuration, I/O, and memory spaces of a Controller in the NVM Subsystem; and
c) Management Interface specific commands for inventorying, configuring, and monitoring of the NVM Subsystem.

The Command Messages supported by an NVM Subsystem are dependent on the mechanism used to send the NVMe-MI Message (i.e., in-band tunneling mechanism or out-of-band mechanism) and whether the NVM Subsystem is contained within an NVMe Storage Device or NVMe Enclosure.

When using the in-band tunneling mechanism, the architecture and behavior of an NVM Subsystem is as defined by the NVM Express Base Specification with extensions defined by this specification. The remainder of this section describes the architecture and behavior of an NVM Subsystem when the out-of-band mechanism is used.

The PCIe ports and SMBus/I2C port of an NVM Subsystem may optionally each contain a single NVMe Management Endpoint (hereafter referred to as simply Management Endpoint). A Management Endpoint is an MCTP endpoint that is the terminus and origin of MCTP packets/messages and is responsible for implementing the MCTP Base Protocol, processing MCTP Control Messages, and internal routing of Command Messages. Each Management Endpoint in an NVM Subsystem 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.

Management Interface Request Messages and Response Messages are transported as MCTP messages with the Message Type set to NVM Express Management Messages over MCTP (refer to the MCTP IDs and Codes specification). All out-of-band mechanism Request Messages originate with the Management Controller and result in a Response Message from a Management Endpoint.

Each Management Endpoint advertises the unique set of capabilities supported by that Management Endpoint. All Management Endpoints may support the same commands even though PCIe ports are full duplex with much higher data rates than SMBus (i.e., both SMBus/I2C and PCIe VDM are capable of providing the same functionality).

Each NVMe Controller in the NVM Subsystem shall provide an NVMe Controller Management Interface (hereafter referred to as simply Controller Management Interface). The Controller Management Interface processes Controller operations on behalf of any Controller (in-band tunneling mechanism) or Management Endpoint (out-of-band mechanism) in the NVM Subsystem. Controllers or Management Endpoints may route commands to any NVMe Controller in the NVM Subsystem. A Controller Management Interface logically processes one operation at a time. A Controller Management Interface is not precluded from processing two or more operations in parallel; however, there shall always be an equivalent pattern of sequential operations with the same results.

Figure 3 illustrates an example NVM Subsystem. The NVM Subsystem contains a single Controller and there is a Management Endpoint associated with the PCIe port.

Figure 3: NVM Subsystem Associated with Single PCIe Port

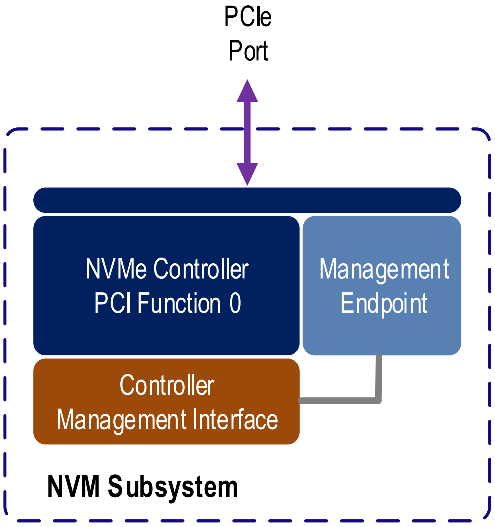
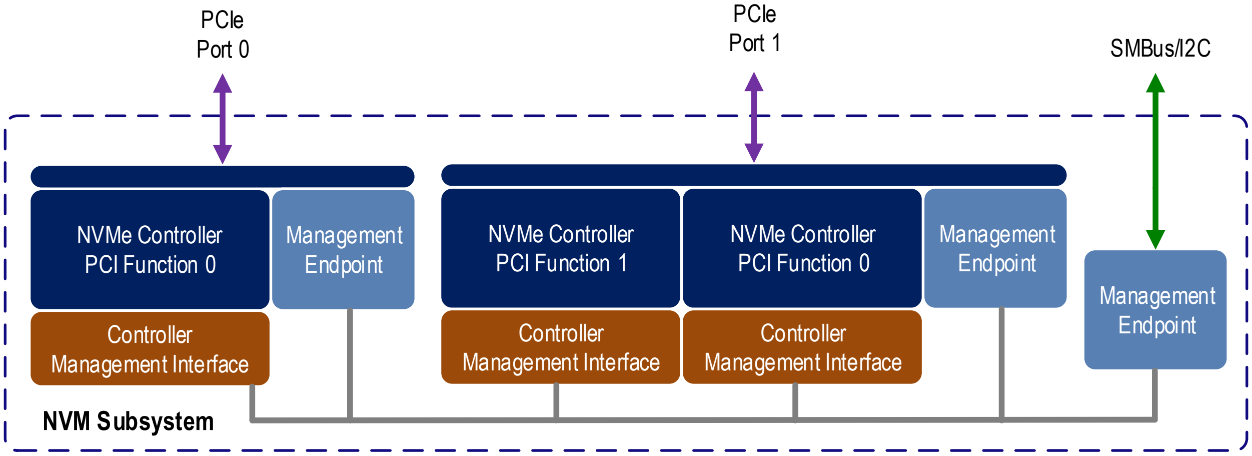


Figure 4 illustrates an example NVM Subsystem that is associated with a dual ported PCIe SSD. The NVM Subsystem contains one Controller associated with PCIe Port 0 and two Controllers associated with PCIe Port 1. There is a Management Endpoint associated with the each PCIe port and the SMBus/I2C port. Since the NVM Subsystem contains a Management Endpoint, all Controllers have an associated Controller Management Interface.

Figure 4: NVM Subsystem with Dual Ported PCIe Ports and an SMBus/I2C Port


PCIe Port 0
PCIe Port 1
SMBus/I2C
NVMe Controller PCI Function 0
Management Endpoint
NVMe Controller PCI Function 1
NVMe Controller PCI Function 0
Management Endpoint
Management Endpoint
Controller Management Interface
Controller Management Interface
Controller Management Interface
NVM Subsystem

1.5 NVMe Storage Device Architectural Model

The architectural model for NVMe Storage Devices that support the in-band tunneling mechanism follows the architectural model defined in the NVM Express Base Specification.

An NVMe Storage Device that implements the out-of-band mechanism but not the in-band tunneling mechanism defined in this specification consists of zero or more NVM Subsystems. An NVMe Storage Device that implements the in-band tunneling mechanisms defined in this specification consists of one or more NVM Subsystems. Each NVM Subsystem that implements the out-of-band mechanism includes one or more Management Endpoints.

An NVMe Storage Device that is a Field-Replaceable Unit (FRU) is a physical component, device, or assembly that is able to be removed and replaced (e.g., by an end user or technician) without having to replace the entire system in which an NVMe Storage Device 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 Base 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 (refer to Figure 5 and Figure 6). 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 (refer to Figure 7). 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 (refer to Figure 8) or Expansion Connectors. NVMe Storage Devices may contain SMBus/I2C Muxes (refer to Figure 8) that connect to one or more NVM Subsystems or Expansion Connectors.

This specification defines 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. VPD is accessible over any port that supports the out-of-band mechanism or in-band tunneling mechanism. If the NVMe Storage Device has an SMBus/I2C port, then the VPD is accessible using the access mechanism over I2C as defined in the IPMI Platform Management FRU Information Storage Definition.

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 accessible via the Upstream Connector describes the entire NVMe Storage Device (refer to section 8.2 for more information). The contents of these additional FRU Information Devices is out of scope for this specification.

Figure 5 illustrates an NVMe Storage Device that is a single-port PCIe SSD with the FRU Information Device implemented by the NVM Subsystem.

Figure 5: Single-Port PCIe SSD

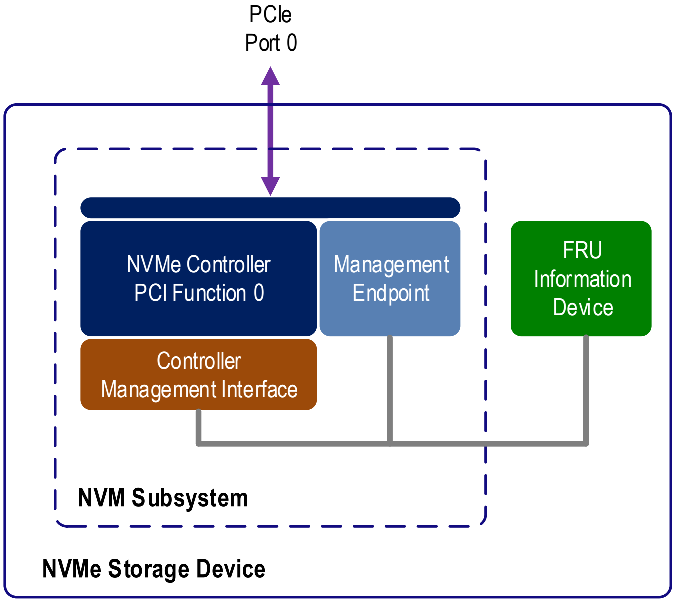
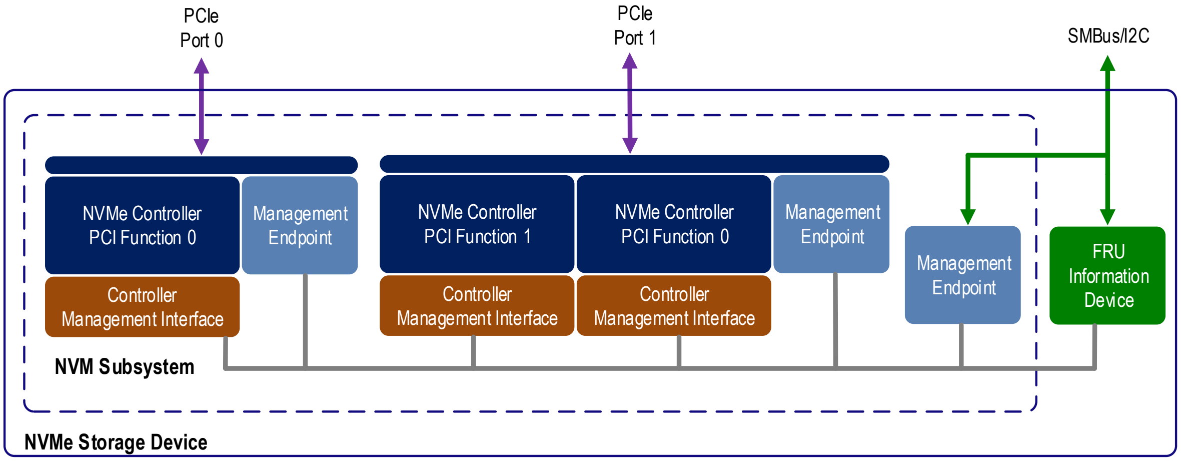


Figure 6 illustrates an NVMe Storage Device that is a dual-port PCIe SSD with an SMBus/I2C port and a FRU Information Device implemented using a Serial EEPROM.

Figure 6: 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 7. 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. As defined by Figure 16, 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 7: NVMe Storage Device with Expansion Connectors (i.e., a Carrier)

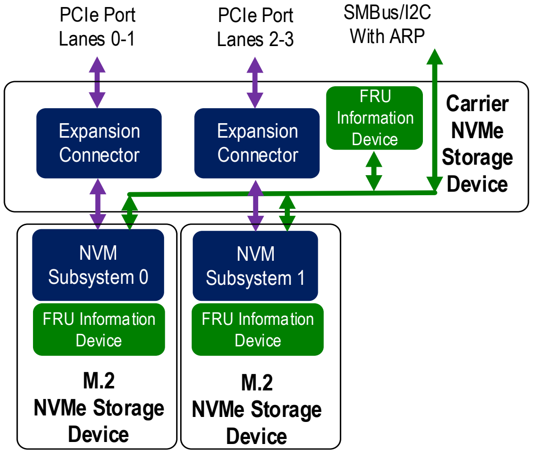
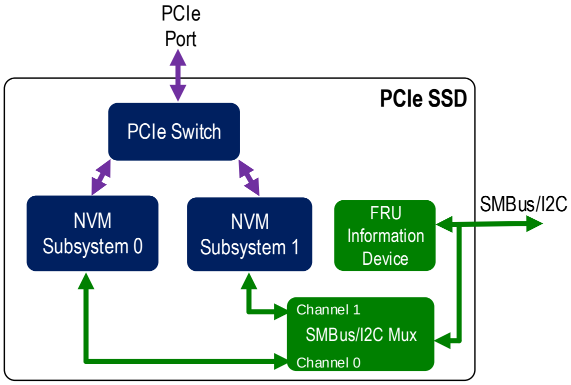


Figure 8 shows an NVMe Storage Device that contains two NVM Subsystems implemented using soldered down ball grid array (BGA) packages and a FRU Information Device at address A6h. An NVMe Storage Device without Expansion Connectors that implements an SMBus/I2C port always contains a FRU Information Device 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 selected NVM Subsystem. The FRU Information Device contains the details necessary to configure the SMBus/I2C Mux.

Figure 8: NVMe Storage Device with two NVM Subsystems and an SMBus/I2C Mux



1.6 NVMe Enclosure Architectural Model

An NVMe 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

NVMe Enclosure itself and/or may be contained in NVMe Storage Devices FRUs that connect to the NVMe Enclosure through one or more NVMe Enclosure slots. An NVMe Enclosure contains one or more NVM Subsystems. NVM Subsystems that are part of an NVMe Enclosure may support just the in-band tunneling mechanism, just the out-of-band mechanism, or both.

An NVMe Enclosure may contain elements that support operation of the NVMe Enclosure (e.g., power supplies, fans, locks, temperature sensors, current sensors, and voltage sensors). An NVMe Enclosure may also contain displays and/or indicators that indicate the state of the NVMe Enclosure (e.g., state of elements, NVM Subsystems, or RAID volumes) and/or NVMe Storage Devices that connect to the NVMe Enclosure. Some of the elements that make up an NVMe Enclosure may be removable and replaceable while the NVMe 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 NVMe Enclosure and a SCSI enclosure are similar and the capabilities required to manage elements of an NVMe Enclosure and a SCSI enclosure are similar. Thus, this specification 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). This specification uses these same control and status diagnostic pages but transfers them using the SES Send and SES Receive commands. this specification supports only the standalone Enclosure Services Process model as defined in SES.

A Requester manages an NVMe 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 Requester to send SES control type diagnostic pages to modify the state of the NVMe Enclosure. The SES Receive command provides the functionality of the SES-3 SCSI RECEIVE DIAGNOSTIC RESULTS command and is used by a Requester to retrieve SES status type diagnostic pages that contain various status and warning information available from the NVMe Enclosure.

Refer to SES-3 for a list and description of SES control type diagnostic pages and SES status type diagnostic pages. The mapping of bytes in SES pages to NVMe-MI Request and Response Data is one-to-one where byte x of the SES page maps to byte x in the NVMe-MI Request or Response Data (e.g., byte zero of the SES control type diagnostic page corresponds to byte zero of NVMe-MI Request Data). 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 NVMe Enclosure elements.

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

An NVMe Enclosure may contain of one or more Subenclosures (refer to SES-3). Each Subenclosure is identified by an 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 NVMe 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 NVMe Enclosure services information for all Subenclosures. Refer to SES-3 for more information.

Associated with each NVMe Enclosure slot is an SES element that may be used to manage the slot. Refer to SES-3 for more information.

Figure 9 illustrates an example NVMe Enclosure that contains one NVM Subsystem. This NVMe Enclosure has multiple ports that Requesters may use to communicate with the NVMe Enclosure. It also has multiple slots that are used to connect NVMe Storage Devices to the NVMe Enclosure (e.g., PCIe). The mapping of NVMe 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 NVMe Enclosure shall contain one or more NVM Subsystems used for Enclosure Management. The NVMe Enclosure in this example may be managed using the out-of-band mechanism via the Responder (Management Endpoint in Figure 9) or using the in-band tunneling mechanism via the NVMe Controller.

Figure 9: Example NVMe Enclosure

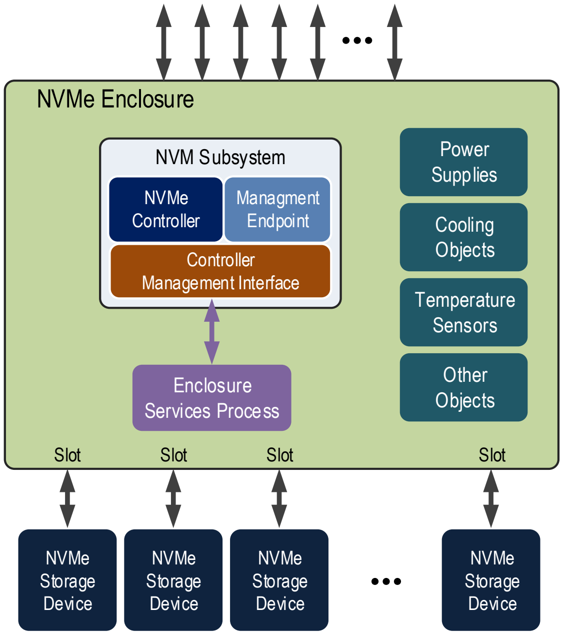


Figure 10 illustrates an example NVMe Enclosure that contains multiple NVM Subsystems and no NVMe Storage Devices. This may represent a software storage appliance. The NVM Subsystems and Controllers contained within these NVM Subsystems may be real or emulated in software. Not all Controllers within these NVM Subsystems are required 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 support for the in-band tunneling mechanism; and others provide access to Namespaces and support for the out-of-band mechanism.

Figure 10: Example NVMe Enclosure with Multiple NVM Subsystems

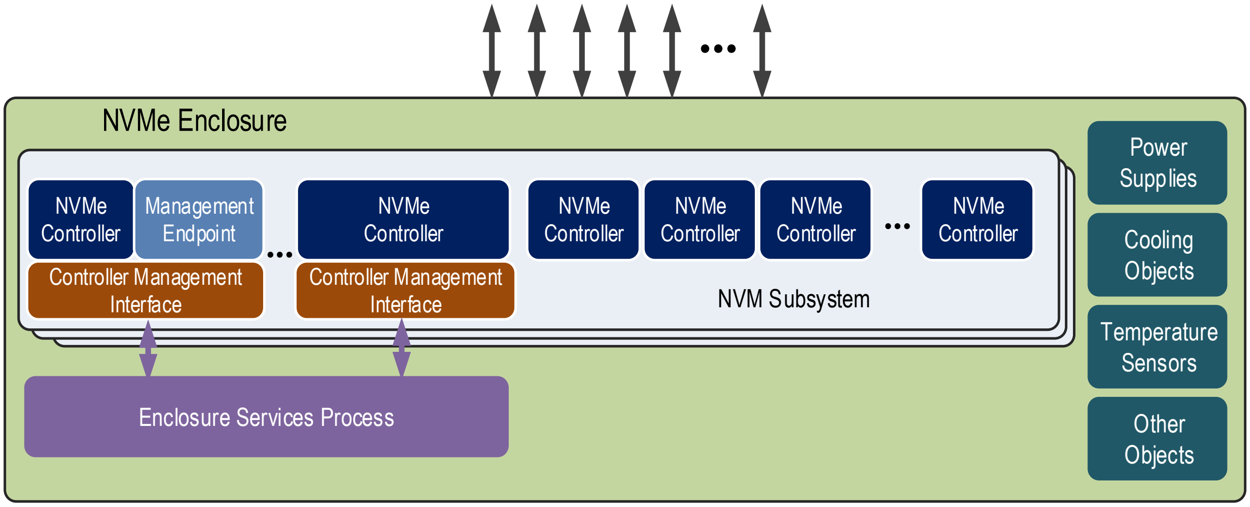


Figure 11 shows an Enclosure that supports two Enclosure Services Processes. Elements of the NVMe Enclosure may be accessible by one or both Enclosure Services Processes. The coordination of access to elements by multiple Enclosure Services Processes is outside the scope of this specification.

Figure 11: Example NVMe Enclosure with Multiple Enclosure Services Processes

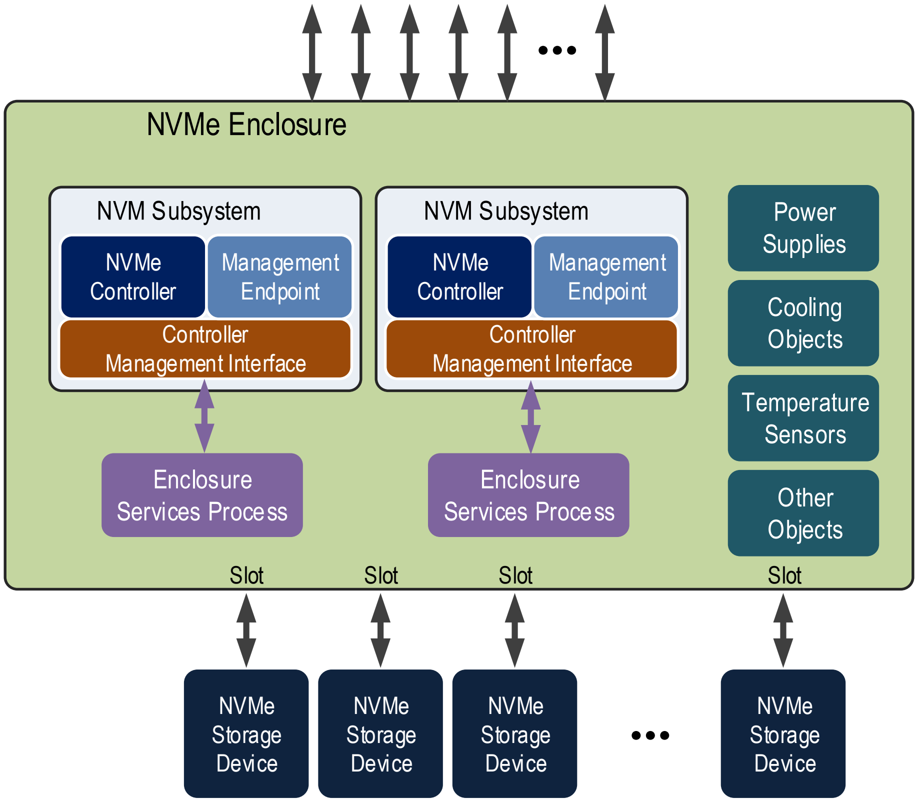
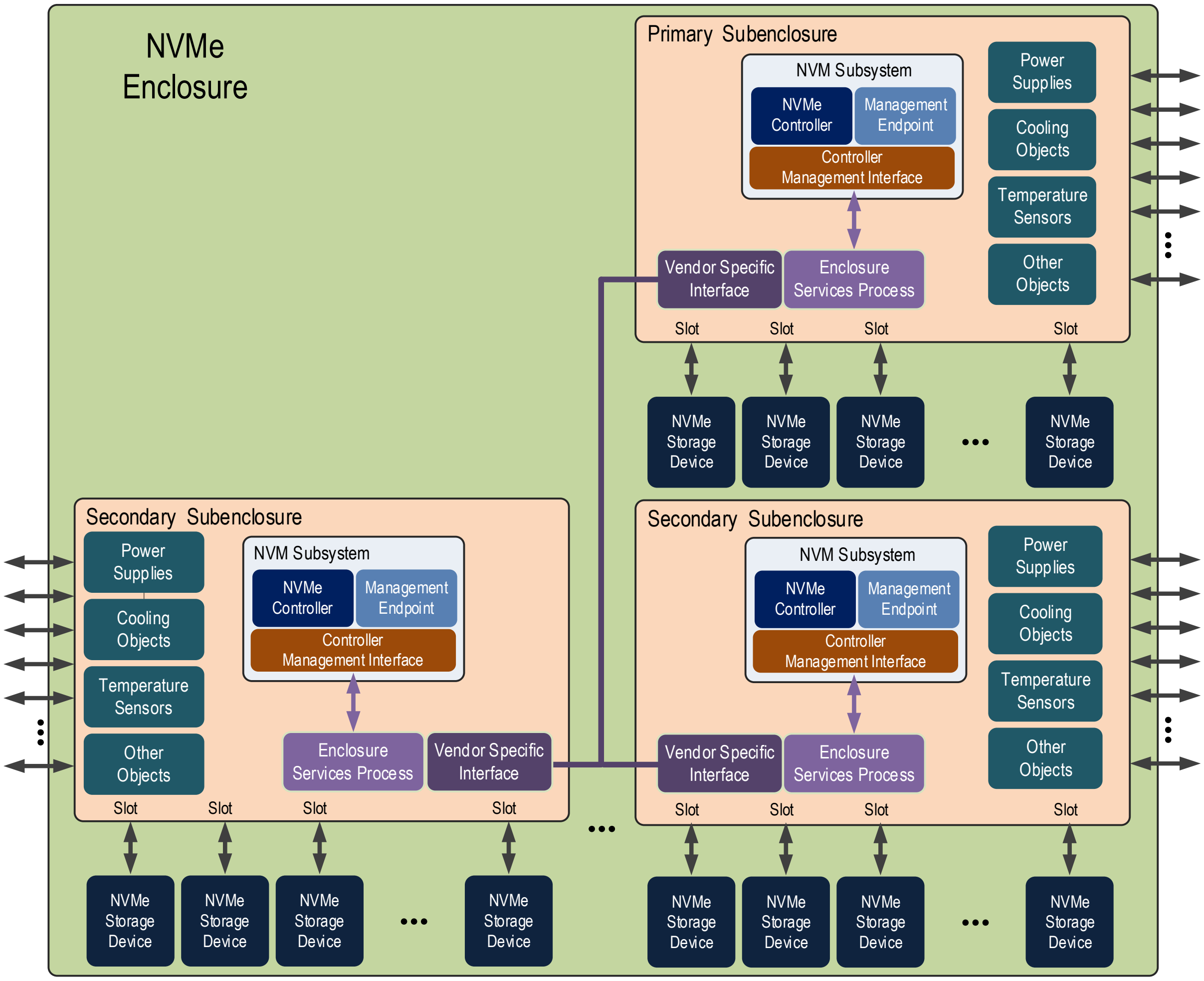


Figure 12 shows an NVMe Enclosure that consists of multiple Subenclosures. Each Subenclosure in this example contains an Enclosure Services Process. NVMe Enclosure services information from Subenclosures is combined into a single set of SES diagnostic pages by the primary Subenclosure. A Subenclosure identifier is used to distinguish from which Subenclosure the information was obtained. Refer to SES-3 for more information. A primary Subenclosure may access NVMe 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 12: Example NVMe Enclosure with Subenclosures



Certain NVMe Enclosure behaviors are managed by setting controls and testing status of elements within an NVMe Enclosure. An Enclosure Services Process may monitor a variety of warning and error conditions. These conditions may be communicated to the Requester through polling by the Requester (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 13. The asynchronous event notification reporting mechanism described in SES-3 is not supported by this specification.

Figure 13: Mapping of SES-3 Sense Keys and Additional Sense Codes to Response Message Status

Response Message Status Values	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

1.7 Conventions

Hardware shall return 0h for all bits, fields, and registers that are marked as reserved. The Requester should not rely on a value of 0h being returned as future revisions of this specification may contain non-zero values. The Requester should write all reserved bits and registers with the value of 0h. Future revisions of this specification may rely on a 0h value being written for backward compatibility.

Hexadecimal (i.e., base 16) numbers are written with a lower case "h" suffix (e.g., 0FFFh, 80h). Hexadecimal numbers larger than eight digits are represented with an underscore character dividing each group of eight digits (e.g., 1E_DEADBEEFh).

Binary (i.e., base 2) numbers are written with a lower case "b" suffix (e.g., 1001b, 10b). Binary numbers larger than four digits are written with an underscore character dividing each group of four digits (e.g., 1000_0101_0010b).

All other numbers are decimal (i.e., base 10). A decimal number is represented in this specification by any sequence of digits consisting of only the Western-Arabic numerals 0 to 9 not immediately followed by a lower-case b or a lower-case h (e.g., 175). This specification uses the following conventions for representing decimal numbers:

a) the decimal separator (i.e., separating the integer and fractional portions of the number) is a period;
b) the thousands separator (i.e., separating groups of three decimal digits in a portion of the number) is a comma;
c) the thousands separator is used in only the integer portion of a number and not the fractional portion of a number; and
d) the decimal representation for a year does not include a comma (e.g., 2018 instead of 2,018).

SMBus/I2C addresses are written as 8-bit hex values where bits 7:1 contain the 7-bit SMBus/I2C address and bit 0 is cleared to '0'.

When a register field is referred to in the document, the convention used is "Property Symbol.Field Symbol" (e.g., the Controller Status (CSTS) register Shutdown Status (SHST) field is referred to by the name CSTS.SHST). If the register field is an array of bits, the field is referred to as "Property Symbol.Field Symbol (array offset to element)". When a sub-field is referred to in the document, the convention used is "Property Symbol.Field Symbol.Sub Field Symbol".

A 0's based value is a numbering scheme for which the number 0h represents a value of 1h, 1h represents 2h, 2h represents 3h, etc. In this numbering scheme, there is not a method for specifying the value of 0h.

Values in this specification are 1-based (i.e., the number 1h represents a value of 1h, 2h represents 2h, etc.) unless otherwise specified.

Some parameters are defined as a string of ASCII or UTF-8 characters. ASCII strings shall contain only code values 20h to 7Eh. UTF-8 is backwards compatible with ASCII encoding and supports additional characters with variable length encoding. For the string "Copyright", the character "C" is the least-significant byte, the character "o" is the second byte, etc. The string is left justified and shall be padded with spaces (ASCII character 20h) to the right if necessary.

A range of numeric values is represented in this specification in the form "a to z", where a is the first value included in the range, all values between a and z are included in the range, and z is the last value included in the range (e.g., the representation "0h to 3h" includes the values 0h, 1h, 2h, and 3h).

Size values are shown in binary units or decimal units. The symbols used to represent these values are as shown in Figure 14.

Figure 14: Decimal and Binary Units

Decimal		Binary	
Symbol	**Power (base-10)**	**Symbol**	**Power (base-2)**
kilo / k	10^3	kibi / Ki	2^{10}
mega / M	10^6	mebi / Mi	2^{20}
giga / G	10^9	gibi / Gi	2^{30}
tera / T	10^{12}	tebi / Ti	2^{40}
peta / P	10^{15}	pebi / Pi	2^{50}
exa / E	10^{18}	exbi / Ei	2^{60}
zetta / Z	10^{21}	zebi / Zi	2^{70}
yotta / Y	10^{24}	yobi / Yi	2^{80}

Implementation Specific (Impl Spec) – the Responder has the freedom to choose its implementation.

Hardware Initialize (HwInit) – The default value is dependent on Responder and system configuration. The value is initialized at reset (e.g., by an expansion ROM, or in the case of integrated devices, by a platform BIOS).

1.8 Definitions

1.8.1 Carrier

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

1.8.2 Command Message

A type of Request Message that contains an NVMe Admin Command, PCIe Command, or NVMe-MI Command.

1.8.3 Command Slot

A logical target within a Management Endpoint where a Management Controller sends a Request Message. Each Management Endpoint has exactly two Command Slots.

1.8.4 Control Primitive

Single-packet Request Messages sent from a Management Controller to a Management Endpoint to affect the servicing of a previously issued Command Message or get the state of a Command Slot and Management Endpoint. Control Primitives are applicable only in the out-of-band mechanism and are prohibited in the in-band tunneling mechanism.

1.8.5 NVMe Controller (Controller)

Refer to the NVM Express Base Specification.

1.8.6 NVMe Controller Management Interface (Controller Management Interface)

An interface associated with each NVMe Controller in the NVM Subsystem that is responsible for processing management operations on behalf of a Management Endpoint.

1.8.7 Enclosure Management

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

1.8.8 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.

1.8.9 Expansion Connector

A connector that allows an NVMe Storage Device FRU or cable to be attached or removed from a Carrier. Expansion Connectors may be empty or populated. A connector to a non-removable NVMe Storage Device is not an Expansion Connector.

1.8.10 Field-Replaceable Unit (FRU)

A physical component, device, or assembly in a system that is able to be removed and replaced (e.g., by an end user or technician) 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 section 1.8.19).

1.8.11 FRU Information Device

A logical or physical device used to hold the VPD. A FRU Information Device may be implemented in a variety of ways (e.g., a serial EEPROM, one-time programmable memory in silicon, etc.).

1.8.12 In-Band

Per the Management Component Transport Protocol (MCTP) Overview White Paper, in-band management is management that operates with the support of hardware components that are critical to and used by the operating system. The in-band communication path defined by this specification is via the NVMe Admin Queue using the NVMe-MI Send and NVMe-MI Receive commands from host software to an NVMe Controller. Refer to the NVM Express Base Specification and section 4.3 of this specification for additional details on the NVMe-MI Send and NVMe-MI Receive commands.

1.8.13 Management Controller

A device (e.g., Baseboard Management Controller (BMC)) responsible for platform management that uses the NVM Express Management Interface to communicate to Management Endpoints.

1.8.14 Management Endpoint or NVMe Management Endpoint

An MCTP endpoint associated with an NVM Subsystem (e.g., an NVMe SSD or NVMe Enclosure) that is the terminus and origin of MCTP packets/messages and which processes Request Messages and transmits Response Messages.

1.8.15 Management Endpoint Buffer

An intermediate buffer defined by this specification to allow servicing out-of-band NVMe-MI Messages that have a Message Body that is larger than the 4,224-byte limit that is specified by the NVMe Management Messages over MCTP Binding Specification.

1.8.16 NVMe 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 Device FRUs. An NVMe Enclosure contains one or more NVM Subsystems and one or more Enclosure Services Processes.

1.8.17 NVMe Processing

NVMe command processing as defined by the NVM Express Base Specification. The term NVMe Processing is used to distinguish command processing as defined by the NVM Express Base Specification from the Command Message processing defined by this specification (refer to section 1.8.24).

1.8.18 NVMe Storage Device

A logical or physical component, device, or assembly that contains at least one NVM Subsystem or Expansion Connector, at least one Upstream Connector, and at least one FRU Information Device. An NVMe Storage Device that implements the out-of-band mechanism contains at least one Management Endpoint and a Controller Management Interface per Controller. An NVMe Storage Device contains zero or more PCIe switches and SMBus/I2C Muxes. An NVMe Storage Device shall comply with the NVM Express Base Specification. In this specification, NVMe Storage Devices shall also comply with this specification.

1.8.19 NVMe Storage Device FRU

An NVMe Storage Device that is able to be removed and replaced (e.g., by an end user or technician) 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.

1.8.20 NVMe Subenclosure (Subenclosure)

A portion of an NVMe Enclosure accessed through a primary NVMe Enclosure’s Enclosure Services Process. Refer to SCSI Enclosure Services - 3 (SES-3) for more information.

1.8.21 NVMe-MI Message

A type of MCTP Message that is defined by this specification in sections 3.1 and 4.1. See the MCTP IDs and Codes specification and the NVMe Management Messages over MCTP Binding Specification for more details on this type of MCTP Message (note that NVMe-MI Messages are referred to as NVM Express Management Messages over MCTP in these specifications).

1.8.22 NVM Subsystem

This specification extends the definition of an NVM Subsystem defined in the NVM Express Base Specification (e.g., by adding a Management Endpoint, Controller Management Interface, etc.). NVMe Enclosures and NVMe Storages devices that are not Carriers have one or more NVM Subsystems. Carriers have zero or more NVM Subsystems.

1.8.23 Out-of-Band

Per the Management Component Transport Protocol (MCTP) Overview White Paper, out-of-band management is management that operates with hardware resources and components that are independent of the operating system's control. The out-of-band communication paths supported by this specification are via MCTP over SMBus/I2C or MCTP over PCIe VDM from a Management Controller to a Management Endpoint. In addition, this specification supports the out-of-band access mechanism defined by the IPMI Platform Management FRU Information Storage Definition specification for accessing a FRU Information Device from a Management Controller over SMBus/I2C.

1.8.24 Process

This is the state when a Command Message is processed. Processing of a Command Message consists of checking for errors with the Command Message and performing the actions specified by the Command Message. This state is applicable in both the out-of-band mechanism and the in-band tunneling mechanism. Refer to section 4.2 for additional details on the Process state in the out-of-band mechanism. Refer to section 4.3 for additional details on the Process state in the in-band tunneling mechanism.

This specification uses the terms process/processing/processed to refer to actions performed in the Process state. These terms are distinct from the NVMe Processing term used to describe NVMe command processing as defined by the NVM Express Base Specification (refer to section 1.8.17 in this specification).

1.8.25 Request Message

An NVMe-MI Message originating from a Requester. A Request Message may be a Command Message or a Control Primitive. Request Messages may be used in both the out-of-band mechanism and the in-band tunneling mechanism.

1.8.26 Requester

The entity that sends Request Messages and receives Response Messages. For the out-of-band mechanism, the Requester is a Management Controller. For the in-band tunneling mechanism, the Requester is host software.

1.8.27 Responder

The entity that receives Request Messages and sends back Response Messages. For the out-of-band mechanism, the Responder is a Management Endpoint. For the in-band tunneling mechanism, the Responder is an NVMe Controller.

1.8.28 Response Message

An NVMe-MI Message originating from a Responder in response to a Request Message. Response Messages may be used in both the out-of-band mechanism and the in-band tunneling mechanism.

1.8.29 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 zero or more downstream channels to the upstream channel. Each downstream channel may be connected to devices with SMBus/I2C ports. This multiplexer permits multiple devices to use the same SMBus/I2C addresses if they are on separate channels.

1.8.30 Upstream Connector

A connector on the NVMe Storage Device or NVMe Enclosure to which a Requester 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.

1.8.31 VPD or Vital Product Data

Field-Replaceable Unit (FRU) Information which is stored in a FRU Information Device. This specification defines a standard VPD format for NVMe Storage Devices.

1.9 Keywords

Several keywords are used to differentiate between different levels of requirements.

1.9.1 mandatory

A keyword indicating items to be implemented as defined by this specification.

1.9.2 may

A keyword that indicates flexibility of choice with no implied preference.

1.9.3 obsolete

A keyword indicating functionality that was defined in a previous version of this specification that has been removed.

1.9.4 optional

A keyword that describes features that are not required by this specification. However, if any optional feature defined by the specification is implemented, the feature shall be implemented in the way defined by the specification.

1.9.5 R

“R” is used as an abbreviation for “reserved” when the figure or table does not provide sufficient space for the full word “reserved”.

1.9.6 reserved

A keyword referring to bits, bytes, words, fields, and opcode values that are set-aside for future standardization. Their use and interpretation may be specified by future extensions to this or other specifications. A reserved bit, byte, word, field, or register shall be cleared to 0h, or in accordance with a future extension to this specification. The recipient of a Request Message or register write is not required to check the value of reserved bits, bytes, words, or fields. Receipt of reserved coded values in defined fields in Request Messages shall be reported as an error. Writing a reserved coded value into a Controller register field produces undefined results.

1.9.7 shall

A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to the specification.

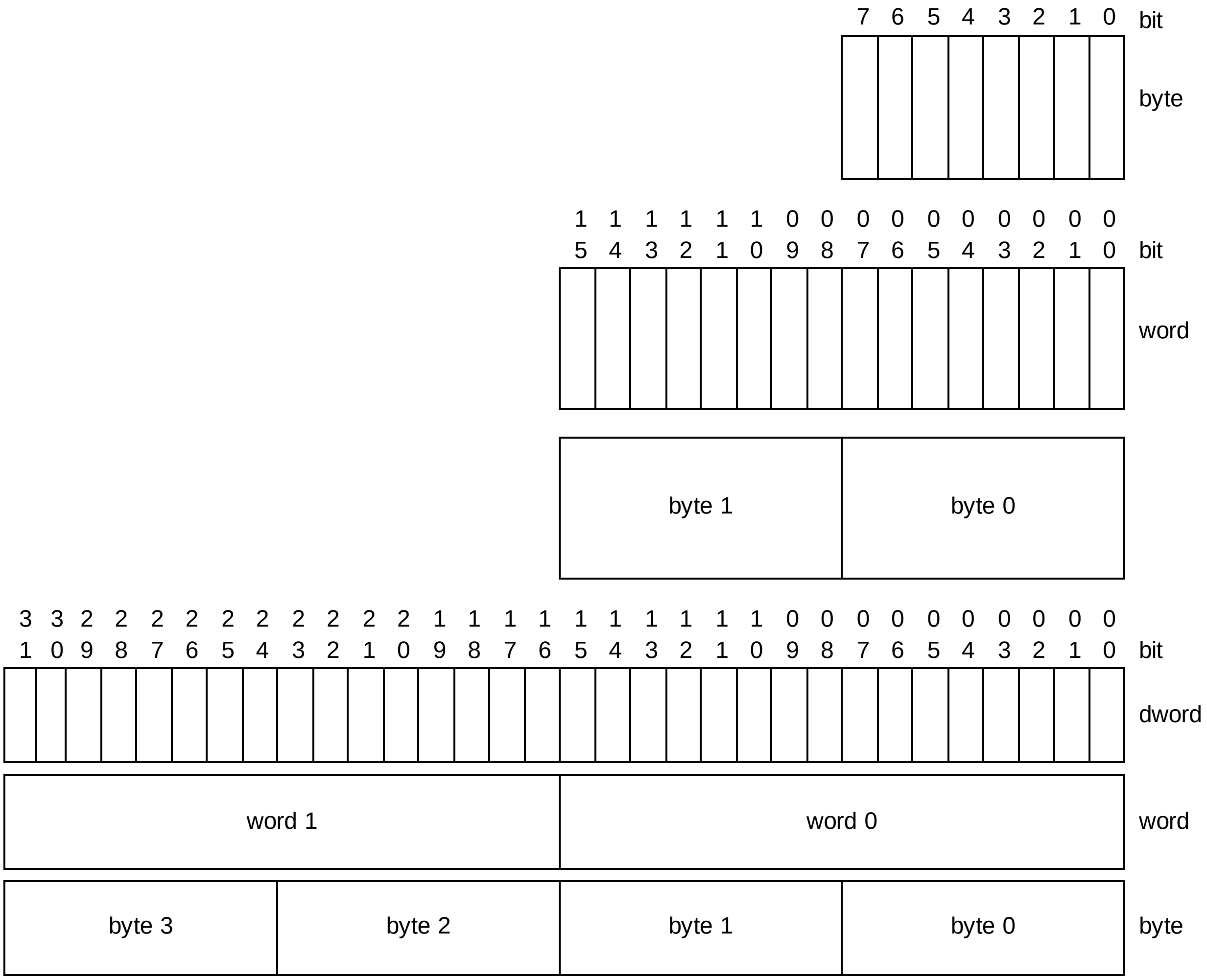
1.9.8 should

A keyword indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase “it is recommended”.

1.10 Byte, Word, and Dword Relationships

Figure 15 illustrates the relationship between bytes, words, and dwords. Unless otherwise stated, this specification specifies data in a little-endian format.

Figure 15: Byte, Word, and Dword Relationships



1.11 References

I2C Bus specification, revision 6.0. Available from https://www.i2c-bus.org

IPMI Platform Management FRU Information Storage Definition 1.0, Version 1.3. Available from https://www.intel.com.

INCITS 518-2017 Information Technology – SCSI Enclosure Services – 3 (SES-3). Available from https://webstore.ansi.org/.

MCTP Base Specification (DSP0236), version 1.3.1. Available from https://www.dmtf.org.

MCTP IDs and Codes (DSP0239), version 1.7.0. Available from https://www.dmtf.org.

MCTP Overview White Paper (DSP2016), version 1.0.0. Available from https://www.dmtf.org.

MCTP PCIe VDM Transport Binding Specification (DSP0238), version 1.1.0. Available from https://www.dmtf.org.

MCTP SMBus/I2C Transport Binding Specification (DSP0237), version 1.2.0. Available from https://www.dmtf.org.

NVM Express Base Specification, revision 2.0. Available from https://www.nvmexpress.org.

NVM Express Key Value Command Set Specification, revision 1.0. Available from https://www.nvmexpress.org.

NVM Express NVM Command Set Specification, revision 1.0. Available from https://www.nvmexpress.org.

NVM Express NVMe over PCIe Transport Specification, revision 1.0. Available from https://www.nvmexpress.org.

NVM Express Zoned Namespace Command Set Specification, revision 1.1. Available from https://www.nvmexpress.org.

NVMe™ (NVM Express™) Management Messages over MCTP Binding specification (DSP0235), revision 1.0.1. Available from https://www.dmtf.org.

PCI Express Base Specification, revision 5.0. Available from https://www.pcisig.com.

PCI-SIG PCI Express Card Electromechanical Specification, Revision 4.0, Version 1.0. Available from https://www.pcisig.com.

PCI-SIG PCI Express M.2 Specification, Revision 3.0, Version 1.2. Available from https://www.pcisig.com.

PCI-SIG PCI Express SFF-8639 Module Specification, Revision 3.0, Version 1.0. Available from https://www.pcisig.com.

SNIA Native NVMe-oF™ Drive Specification, Version 1.0.1. Available from https://www.snia.org.

SNIA SFF-TA-1001 Universal x4 Link Definition for SFF-8639 Specification, Revision 1.1. Available from https://www.snia.org.

SNIA SFF-TA-1006 Enterprise and Datacenter 1U Short SSD Form Factor (E1.S) Specification, Revision 1.3a. Available from https://www.snia.org.

SNIA SFF-TA-1007 Enterprise and Datacenter 1U Long SSD Form Factor (E1.L) Specification, Revision 1.1. Available from https://www.snia.org.

SNIA SFF-TA-1008 Enterprise and Datacenter 3” SSD Form Factor Specification. Available from https://www.snia.org.

System Management Bus (SMBus) Specification, revision 3.1. Available from http://www.smbus.org.

2 Physical Layer

This section describes the physical layers supported by this specification for NVMe Storage Devices or NVMe Enclosures.

2.1 PCI Express

PCI Express is used as a physical layer in both the out-of-band mechanism and the in-band tunneling mechanism in this specification.

For the out-of-band mechanism, a PCIe port in an NVMe Storage Device or NVMe Enclosure may implement a Management Endpoint. If the PCIe port implements a Management Endpoint, the PCIe port shall support MCTP over PCIe Vendor Defined Messages (VDMs) as specified by the MCTP PCIe VDM Transport Binding Specification.

For the in-band tunneling mechanism, host software issues NVMe Admin Commands (NVMe-MI Send and NVMe-MI Receive) to the NVMe Admin Queue over PCI Express. Refer to the NVM Express Base Specification and section 4.3 of this specification for additional details on the NVMe-MI Send and NVMe-MI Receive commands.

2.2 SMBus/I2C

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

If an NVMe Storage Device or NVMe Enclosure implements an NVM Subsystem with a Management Endpoint associated with an SMBus/I2C port, then that port shall comply to the MCTP SMBus/I2C Transport Binding Specification.

An NVM Subsystem may also support the NVMe Basic Management Command for health and status polling. The NVMe Basic Management Command is defined as an informative technical note in Appendix A, though it is not recommended for new designs.

Figure 16 lists SMBus/I2C elements that are supported on an NVMe Storage Device or NVMe Enclosure. 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 or NVMe Enclosure. 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 a different SMBus/I2C address for the FRU Information Device on NVMe Storage Devices that are Carriers versus non-Carriers avoids SMBus/I2C address conflict when Expansion Connectors are populated with NVMe Storage Devices.

ARP support on SMBus/I2C elements is optional unless multiple SMBus/I2C elements in the NVMe Storage Device or NVMe Enclosure with the same default SMBus/I2C address are present on the same SMBus/I2C channel.

Figure 16: 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. Undefined on NVMe Enclosures.
FRU Information Device	A4h	1010_010xb	Optional	Required on Carriers (i.e., an NVMe Storage Device with *one or more* Expansion Connectors). Undefined on NVMe Enclosures.
SMBus/I2C Management Endpoint	3Ah	0011_101xb	Optional	Required if an NVMe Storage Device or NVMe Enclosure contains an SMBus/I2C Management Endpoint.
SMBus/I2C Mux	E8h	1110_100xb	Optional	For NVMe Storage Devices, 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. Undefined on NVMe Enclosures.
Basic Management Command [2]	D4h	1101_010xb	Optional	For NVMe Storage Devices, not recommended for new designs. Undefined on NVMe Enclosures.

NOTES:
1. The x represents the SMBus/I2C read/write bit.
2. The NVMe Basic Management Command is defined in Appendix A as an informative technical note.

Host platforms expecting to be used with one or more Management Endpoints (e.g., data center platforms and workstations) should isolate SMBus/I2C 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 (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 or NVMe Enclosure.

SMBus/I2C elements that support ARP should be implemented as DSA devices (refer to the SMBus Specification). These devices should not issue “Notify ARP Master” commands.

If ARP is supported by an NVM Subsystem, then all SMBus/I2C elements associated with that NVM Subsystem shall use the SMBus Address Resolution Protocol Unique Device Identifier (UDID) shown in Figure 17. The ARP UDID is a unique identifier. The UDID Vendor ID bits 31:30 allow up to four SMBus/I2C elements to be grouped together with the same NVM Subsystem. The only difference within this group of UDIDs is the most-significant two bits of the Vendor Specific ID. This fact may be used by the Management Controller to associate an SMBus/I2C Management Endpoint with its corresponding FRU Information Device.

If there are multiple NVM Subsystems in an SMBus ARP-capable NVMe Storage Device or NVMe Enclosure, 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.

Clock stretching is allowed by the Management Controller, Management Endpoint, and the FRU Information Device. However, implementations are strongly discouraged from using clock stretching so that communications are more predictable with higher throughput.

When a NACK is received, a Management Endpoint shall follow the MCTP SMBus/I2C Transport Binding Specification for a non-bridge endpoint. The Management Endpoint treats a STOP condition due to excessive SMBus NACKs as an implicit Pause Control Primitive. Refer to section 4.2.1.1.

Figure 17: 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><b>Address Type:</b> This field describes the type of address contained in the device. Refer to the MCTP SMBus/I2C Transport Binding Specification.</td></tr>
<tr><td>5:1</td><td>Reserved</td></tr>
<tr><td>0</td><td><b>PEC Supported:</b> All MCTP transactions shall include a Packet Error Code (PEC) byte. This bit shall be set to ‘1’ to indicate support for PEC.</td></tr>
</table></td></tr>
<tr><td>119:112</td><td>Version and 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><b>UDID Version:</b> This field specifies the UDID version and shall be set to 001b.</td></tr>
<tr><td>2:0</td><td><b>Silicon Revision ID:</b> 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>15:08</td><td>Reserved</td></tr>
<tr><td>07</td><td><b>ZONE:</b> This bit shall be cleared to ‘0’.</td></tr>
<tr><td>06</td><td><b>IPMI:</b> This bit shall be cleared to ‘0’.</td></tr>
<tr><td>05</td><td><b>ASF:</b> This bit shall be set to ‘1’. Refer to the MCTP SMBus/I2C Transport Binding Specification.</td></tr>
<tr><td>04</td><td><b>OEM:</b> This bit shall be set to ‘1'.</td></tr>
<tr><td>03:00</td><td><b>SMBus Version:</b> This field shall be set to 4h for SMBus Version 2.0, or to 5h for SMBus Version 3.0 and 3.1.</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>

Figure 17: SMBus/I2C Element UDID

<table>
<tr><th>Bits</th><th>Field</th><th>Description</th></tr>
<tr><td>31:00</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 or NVMe Enclosure.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:30</td><td><b>UDID Type:</b> This field distinguishes which NVM Subsystem that implements multiple SMBus elements is providing the UDID. Note that Management Controllers implemented to versions prior to NVMe-MI 1.1 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>Vendor Specific Devices</td></tr>
</table></td></tr>
<tr><td>29:00</td><td><b>UDID Device ID:</b> 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>

2.3 Error Handling

Physical layer errors are handled as specified by the corresponding physical layer specification and MCTP transport binding specification. This specification does not require any physical layer specific error handling requirements beyond those outlined in the MCTP transport binding specifications.

3 Message Transport

This specification defines an interface that supports multiple message transports. The message format is the same for the out-of-band mechanism and the in-band tunneling mechanism as described in section 3.1. The out-of-band message transport is described in section 3.2. The in-band tunneling message transport is described in section 3.3.

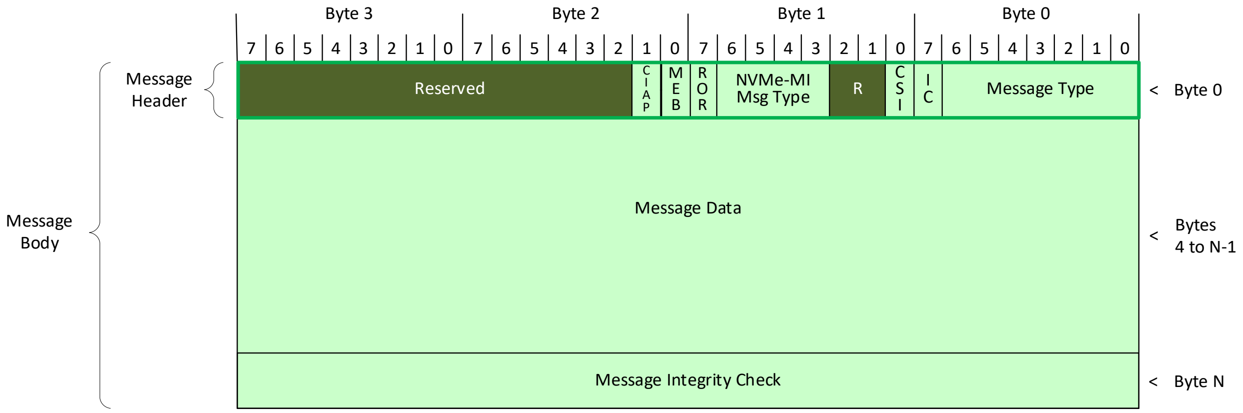
3.1 NVMe-MI Messages

NVMe-MI Messages are used in both the out-of-band mechanism and the in-band tunneling mechanism. The format of an NVMe-MI Message is shown in Figure 18 and Figure 19.

In the out-of-band mechanism, an NVMe-MI Message consists of the payload of one or more MCTP packets. The maximum sized NVMe-MI Message is 4,224 bytes (i.e., 4 KiB + 128 bytes). Refer to the NVMe Management Messages over MCTP Binding Specification. NVMe-MI Messages with lengths greater than 4,224 bytes are considered invalid NVMe-MI Messages. See section 4.2 for details on how NVMe-MI Messages are used in the out-of-band mechanism.

In the in-band tunneling mechanism, NVMe-MI Messages are not split into MCTP packets and the maximum NVMe-MI message size is equal to the Maximum Data Transfer Size (refer to the NVM Express Base Specification). See section 4.3 for details on how NVMe-MI Messages are used in the in-band tunneling mechanism.

Figure 18: NVMe-MI Message



3.1.1 Message Fields

The format of an NVMe-MI Message consists of a Message Header in the first dword, followed by the Message Data. If the Integrity Check (IC) bit is set to '1', then the NVMe-MI Message ends with the Message Integrity Check as shown in Figure 18.

The Message Header contains a Message Type (MT) field and an Integrity Check (IC) bit 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 NVMe-MI Messages (refer to the MCTP IDs and Codes specification). The Integrity Check (IC) bit indicates whether the NVMe-MI Message is protected by a Message Integrity Check. All NVMe-MI Messages in the out-of-band mechanism shall be protected by a 32-bit CRC computed over the Message Body contents. The IC bit shall be set to '1' in all NVMe-MI

Messages in the out-of-band mechanism. The Integrity Check (IC) bit shall be cleared to ‘0’ in all NVMe-MI Messages in the in-band tunneling mechanism.

The Request or Response (ROR) bit in the Message Header specifies whether the NVMe-MI Message is a Request Message or a Response Message. The NVMe-MI Message Type (NMIMT) field specifies whether the Request Message is a Control Primitive or a specific type of Command Message (refer to Figure 24). The Command Slot Identifier (CSI) bit specifies the Command Slot with which the NVMe-MI Message is associated in the out-of-band mechanism. Refer to section 4.2 for additional information about Command Slots.

The Management Endpoint Buffer (MEB) bit in the Message Header specifies whether Message Data is contained in the associated Message Data field of an NVMe-MI Message or in the Management Endpoint Buffer. This bit should only be set to ‘1’ in Command Messages that support Management Endpoint Buffer operation (i.e., those listed in the Management Endpoint Buffer Supported Command List data structure). If the MEB bit is set to ‘1’ in any other Command Message, then the Management Endpoint shall respond with an Invalid Parameter Error Response with the PEL field indicating the MEB bit.

The Command Initiated Auto Pause (CIAP) bit in the Message Header of a Command Message specifies whether or not the Management Endpoint is automatically paused when a Command Message enters the Process state. A Command Message with the CIAP bit set to ‘1’ shall be treated by the Management Endpoint as if an implicit Pause Control Primitive, as described in section 4.2.1.1, was received in the Process state with the exception that the Management Endpoint shall not transmit a Control Primitive Response Message. The Command Initiated Auto Pause Supported (CIAPS) bit in Figure 95 indicates if the port supports the Command Initiated Auto Pause (CIAP) bit in Command Messages.

Figure 19: NVMe-MI Message Fields

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>0</td><td><b>MCTP Data (MCTPD):</b> This field contains the Message Type and Integrity Check fields as defined by the MCTP Base Specification.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7</td><td><b>Integrity Check (IC):</b> This bit is defined by the MCTP Base Specification and indicates whether the MCTP message is covered by an overall MCTP Message Integrity Check.
All NVMe-MI Messages in the out-of-band mechanism shall be protected by a CRC and thus this bit shall be set to ‘1’ in all out-of-band NVMe-MI Messages.
All NVMe-MI Messages in the in-band tunneling mechanism shall not be protected by a CRC and thus this bit shall be cleared to ‘0’ in all in-band NVMe-MI Messages.</td></tr>
<tr><td>6:0</td><td><b>Message Type (MT):</b> This field is defined by the MCTP Base Specification for the message type. This field shall be set to 4h in all NVMe-MI Messages. Refer to the MCTP IDs and Codes specification.</td></tr>
</table>
</td></tr>
</table>

Figure 19: NVMe-MI Message Fields

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>1</td><td><b>NVMe-MI Message Parameters (NMP):</b> This field contains parameters applicable to the NVMe-MI Message.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7</td><td><b>Request or Response (ROR):</b> This bit indicates whether the message is a Request Message or Response Message. This bit is cleared to ‘0’ for Request Messages. This bit is set to ‘1’ for Response Messages.</td></tr>
<tr><td>6:3</td><td><b>NVMe-MI Message Type (NMIMT):</b> This field specifies the NVMe-MI Message Type. Refer to the sections referenced in the table below for details about each NVMe-MI Message Type and whether they apply to the out-of-band mechanism, the in-band tunneling mechanism, or both.
<table>
<tr><th>Value</th><th>Description</th><th>Reference Section</th></tr>
<tr><td>0h</td><td>Control Primitive</td><td>4.2.1</td></tr>
<tr><td>1h</td><td>NVMe-MI Command</td><td>5</td></tr>
<tr><td>2h</td><td>NVMe Admin Command</td><td>6</td></tr>
<tr><td>3h</td><td>Reserved</td><td>-</td></tr>
<tr><td>4h</td><td>PCIe Command</td><td>7</td></tr>
<tr><td>5h to Fh</td><td>Reserved</td><td>-</td></tr>
</table></td></tr>
<tr><td>2:1</td><td>Reserved</td></tr>
<tr><td>0</td><td><b>Command Slot Identifier (CSI):</b> This bit indicates the Command Slot with which the NVMe-MI Message is associated. For Request Messages this bit indicates the Command Slot with which the Request Message is associated. For Response Messages, this bit indicates the Command Slot associated with the Request Message with which the Response Message is associated. This bit is only applicable to NVMe-MI Messages in the out-of-band mechanism. This bit is reserved for NVMe-MI Messages in the in-band tunneling mechanism.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0b</td><td>Command Slot 0</td></tr>
<tr><td>1b</td><td>Command Slot 1</td></tr>
</table></td></tr>
</table></td></tr>
</table>

Figure 19: NVMe-MI Message Fields

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>2</td><td>
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:2</td><td>Reserved</td></tr>
<tr><td>1</td><td><b>Command Initiated Auto Pause (CIAP):</b> If this bit is set to ‘1’ in a Command Message, the Management Endpoint shall be automatically paused when the Command Message enters the Process state. If this bit is cleared to ‘0’ in a Command Message, the Management Endpoint shall not be automatically paused when the Command Message enters the Process state.
This bit is only valid for Command Messages sent using the out-of-band mechanisms and is reserved for all other types of NVMe-MI Messages. If this bit is set to ‘1’ for any type of NVMe-MI Message received by a Management Endpoint other than a Command Message, then an Invalid Parameter Error Response with the PEL field indicating this bit shall be returned.
If this bit is set to ‘1’ in the in-band tunneling mechanism, then an Invalid Parameter Error Response with the PEL field indicating this bit shall be returned.</td></tr>
<tr><td>0</td><td><b>Management Endpoint Buffer (MEB):</b> This bit indicates whether the Message Data in a Command Message is contained in the Message Data field of this NVMe-MI Message or in the Management Endpoint Buffer. Refer to section 3.1.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0b</td><td>The Message Data is contained in the Message Data of this NVMe-MI Message.</td></tr>
<tr><td>1b</td><td>The Message Data is contained in the Management Endpoint Buffer.</td></tr>
</table>
This bit is only valid for Command Messages sent using the out-of-band mechanism and is reserved for all other types of NVMe-MI Messages. If this bit is set to ‘1’ for any type of NVMe-MI Message received by a Management Endpoint other than a Command Message, then an Invalid Parameter Error Response with the PEL field indicating this bit shall be returned.
If this bit is set to ‘1’ in the in-band tunneling mechanism, then an Invalid Parameter Error Response with the PEL field indicating this bit shall be returned.</td></tr>
</table>
</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
<tr><td>N-1:4</td><td><b>Message Data (DATA):</b> This field contains the NVMe-MI Message payload. The format of this field depends on the NVMe-MI Message Type.</td></tr>
<tr><td>N+3:N</td><td><b>Message Integrity Check (MIC):</b> If the Integrity Check (IC) bit is set to ‘1’, then this field contains a CRC computed over the contents of the NVMe-MI Message. Refer to section 3.1.1.1.
If the IC bit is cleared to ‘0’, then this field is not included in the NVMe-MI Message.
This field is byte aligned.</td></tr>
</table>

3.1.1.1 Message Integrity Check

If the Integrity Check (IC) bit is set to ‘1’, then the Message Integrity Check field contains a 32-bit CRC computed over the contents of the NVMe-MI Message. The 32-bit CRC required by this specification is CRC-32C (Castagnoli) which uses the generator polynomial 1EDC6F41h. The Message Integrity Check is calculated using the following Rocksoft™ Model CRC Algorithm parameters:

Name	:	"CRC-32C"
Width	:	32
Poly	:	1EDC6F41h
Init	:	FFFFFFFFh
RefIn	:	True
RefOut	:	True
XorOut	:	FFFFFFFFh
Check	:	E3069283h

When sending a message, the Message Integrity Check shall be calculated using the following procedure or a procedure that produces an equivalent result:

1. Initialize the CRC register to FFFFFFFFh. This is equivalent to inverting the least-significant 32 bits of the NVMe-MI Message (Dword 0 in Figure 18);
2. Append 32 bits of 0's to the end of the Message Data to allow room for the Message Integrity Check (Dword N in Figure 18). This results in the Message Body shown in Figure 18 with the Message Integrity Check field cleared to 0h;
3. Map the bits in the Message Body from step 2 to the coefficients of the message polynomial M(x). Assume the length of M(x) is Y bytes. Bit 0 of byte 0 in the Message Body is the most-significant bit of M(x), followed by bit 1 of byte 0, on through to bit 7 of byte Y - 1. Note that the bits within each byte are reflected (i.e., bit n of each byte is mapped to bit (7 - n) resulting in bit 7 to bit 0, bit 6 to bit 1, and so on);

Figure 20: Message Integrity Check Example

	Message Body (Length = Y bytes)																								
	Byte 0								**Byte 1**								**…**	**Byte Y - 1**							
M(x) =	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	…	0	1	2	3	4	5	6	7

4. Divide the polynomial M(x) by the generator polynomial 1EDC6F41h to produce the 32-bit remainder polynomial R(x);
5. Reflect each byte of R(x) (i.e., bit n of each byte is mapped to bit (7 - n) resulting in bit 7 to bit 0, bit 6 to bit 1, and so on) to produce the polynomial R′(x);
6. Invert R′(x) to produce the polynomial R″(x); and
7. Store R″(x) in the Message Integrity Check field of the Message Body.

Upon receipt of an NVMe-MI Message, the Message Integrity Check may be validated as follows:

1. Save the received Message Integrity Check;
2. Initialize the CRC register to FFFFFFFFh. This is equivalent to inverting the least-significant 32 bits of the NVMe-MI Message (Dword 0 in Figure 18);
3. Clear the Message Integrity Check field to 0h;
4. Map the bits in the Message Body to the coefficients of the message polynomial M(x) as described in step 3 in the Message Integrity Check calculation procedure above;
5. Divide the polynomial M(x) by the generator polynomial 1EDC6F41h to produce the 32-bit remainder polynomial R(x);
6. Reflect each byte of R(x) (i.e., bit n of each byte is mapped to bit (7 - n) resulting in bit 7 to bit 0, bit 6 to bit 1, and so on) to produce the polynomial R′(x);
7. Invert R′(x) to produce the polynomial R″(x); and
8. Compare R″(x) from step 7 to the Message Integrity Check value saved in step 1. If both values are equal, the Message Integrity Check passes.

Refer to Appendix B for artificial messages and their corresponding Message Integrity Check values.

See section 4.2.1.5 for special requirements on how to construct the Response Message when the Management Controller issues a Replay Control Primitive with a non-zero Response Replay Offset.

3.2 Out-of-Band Message Transport

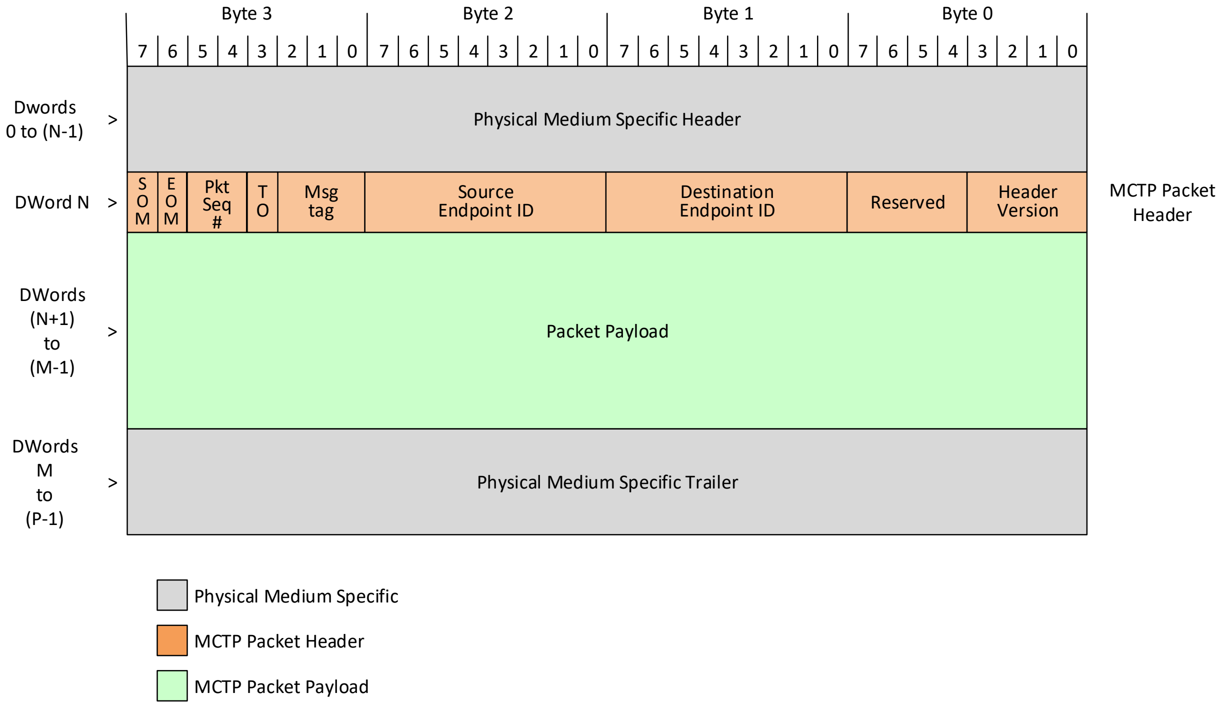
The out-of-band mechanism defined in this specification utilizes MCTP as a reliable in-order message transport between a Management Controller and a Management Endpoint.

This section summarizes the NVMe-MI MCTP packet format. A Management Endpoint compliant to this specification shall implement all required behaviors detailed in the Management Component Transport Protocol (MCTP) Base Specification and corresponding MCTP transport binding specification in addition to the requirements outlined in this specification (e.g., the Message Integrity Check algorithm).

3.2.1 MCTP Packet

In the MCTP Base Specification, the smallest unit of data transfer is the MCTP packet. One or more packets are combined to create an MCTP message. In this specification, the MCTP messages are referred to as NVMe-MI Messages (refer to section 1.8.21). Refer to section 3.2.1.1 for details on how MCTP packets are assembled into NVMe-MI Messages. An MCTP Packet Payload contains at least 1 byte but shall not exceed the negotiated MCTP Transmission Unit Size. The format of an MCTP Packet Payload is shown in Figure 21.

Figure 21: MCTP Packet Format



MCTP specifications use big endian byte ordering while NVM Express specifications use little endian byte ordering. All figures in this specification are illustrated with little endian byte ordering. Note that this pictorial representation does not change the order that bytes are sent out on the physical layer.

The Physical Medium-Specific Header and Physical Medium-Specific Trailer are defined by the MCTP transport binding specification utilized by the port. Refer to the MCTP transport binding specifications.

The Management Component Transport Protocol (MCTP) Base Specification defines the MCTP packet header (refer to DSP0236 for field descriptions). The fields of an MCTP Packet are shown in Figure 22.

Figure 22: MCTP Packet Fields

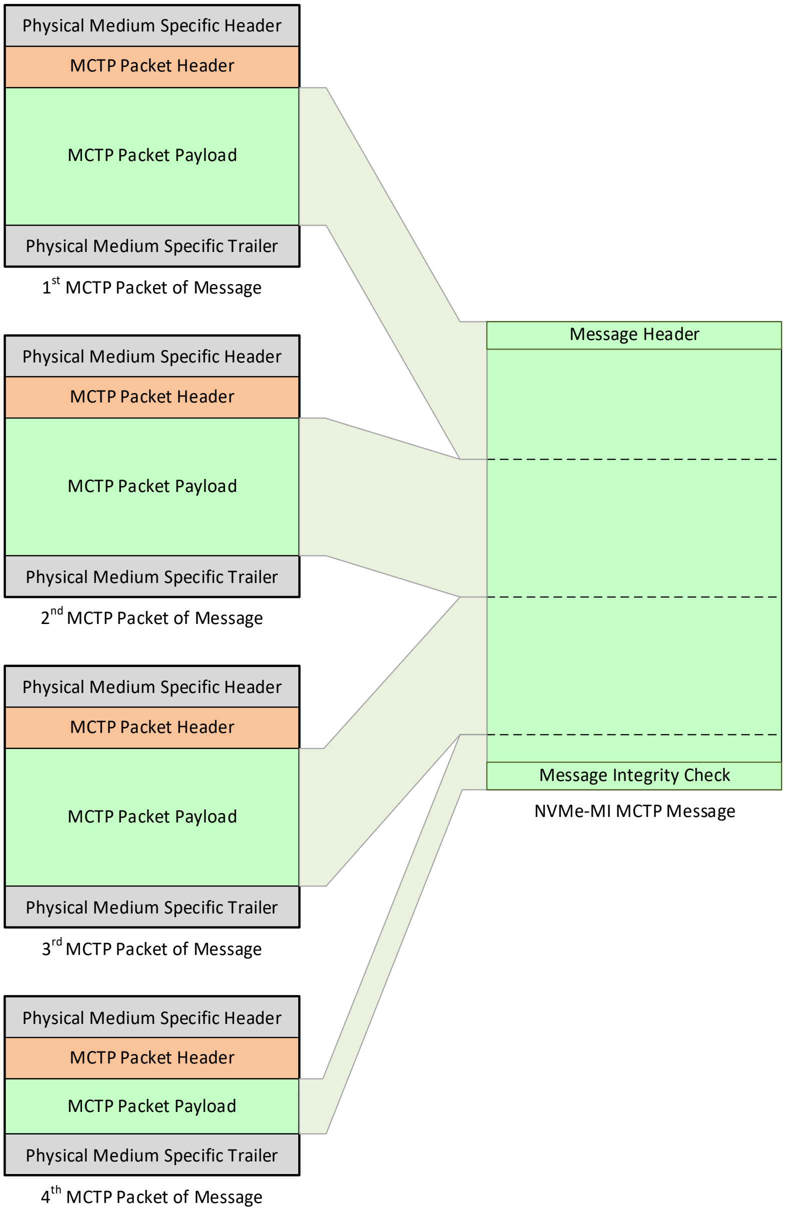
Field Name	Field Size
Medium-Specific Header	varies
Header Version	4 bits
Reserved	4 bits
Destination Endpoint ID	8 bits
Source Endpoint ID	8 bits
Msg tag (Message Tag)	3 bits
TO	1 bit
Pkt Seq #	2 bits
EOM	1 bit
SOM	1 bit
Packet Payload	varies
Medium-Specific Trailer	varies

A compliant Management Endpoint shall implement all MCTP required features defined in the MCTP Base Specification. Optional features may be supported.

3.2.1.1 Packet Assembly into Messages

An NVMe-MI Message may be split into multiple MCTP Packet Payloads and sent as a series of packets. An example NVMe-MI Message whose contents are split across four MCTP packets is shown in Figure 23. Refer to the MCTP Base Specification for packetization and message assembly rules.

Figure 23: NVMe-MI Message Spanning Multiple MCTP Packets


Physical Medium Specific Header
MCTP Packet Header
MCTP Packet Payload
Physical Medium Specific Trailer
1st MCTP Packet of Message
Physical Medium Specific Header
MCTP Packet Header
MCTP Packet Payload
Physical Medium Specific Trailer
2nd MCTP Packet of Message
Physical Medium Specific Header
MCTP Packet Header
MCTP Packet Payload
Physical Medium Specific Trailer
3rd MCTP Packet of Message
Physical Medium Specific Header
MCTP Packet Header
MCTP Packet Payload
Physical Medium Specific Trailer
4th MCTP Packet of Message
Message Header
Message Integrity Check
NVMe-MI MCTP Message

In addition to the requirements outlined in the MCTP Base Specification and MCTP transport binding specifications, this specification has the following additional requirements:

- With the exception of the last packet in a message, the MCTP Transmission Unit size of all packets in a given message shall be equal to the negotiated MCTP Transmission Unit Size;
- The MCTP Transmission Unit size of the last packet in a Request Message or Response Message (i.e., the one with the EOM bit set in the MCTP header) shall be the smallest size required to transfer the MCTP Packet Payload for that Packet with no additional padding beyond any padding required by the physical medium-specific trailer; and

- Once a complete NVMe-MI Message has been assembled, the Message Integrity Check is verified. If the Message Integrity Check passes, then the NVMe-MI Message is processed. If the Message Integrity Check fails, then the NVMe Message is discarded. Refer to section 4.2.

3.2.2 Out-of-Band Error Handling

The Management Endpoint shall drop (silently discard) packets for error conditions as specified in the MCTP Base Specification. Some example conditions which result in discarding packets include unexpected middle or end packets. Silently discarded packets also cause the corresponding bit in the Get State Control Primitive Success Response Fields to be set to '1' (refer to Figure 42).

3.3 In-Band Tunneling Message Transport

The in-band tunneling mechanism in this specification utilizes the NVMe Admin Commands NVMe-MI Send and NVMe-MI Receive as a message transport. Refer to the NVM Express Base Specification and section 4.3 of this specification for additional details on the NVMe-MI Send and NVMe-MI Receive commands.

4 Message Servicing Model

NVMe-MI Messages are used for communication in both the out-of-band and in-band tunneling message servicing models and are described in section 4.1. This specification defines multiple message servicing models. The out-of-band message servicing model is described in section 4.2. The in-band tunneling message servicing model is described in section 4.3.

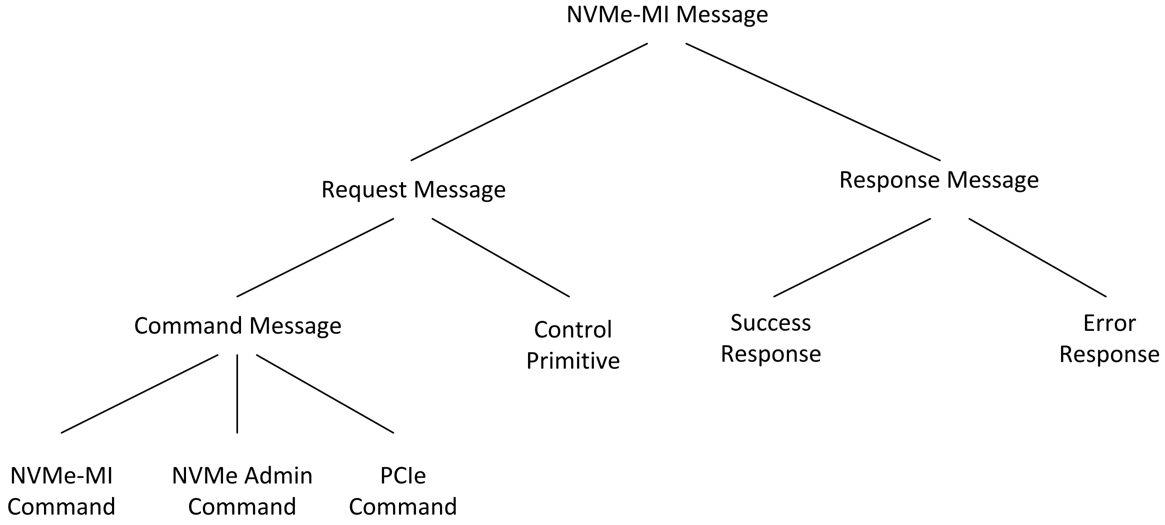
4.1 NVMe-MI Messages

Figure 24 illustrates the taxonomy of NVMe-MI Messages. The two main categories of NVMe-MI Messages are Request Messages and Response Messages. Request Messages are sent by a Management Controller to a Management Endpoint when using the out-of-band mechanism. Request Messages are sent by host software to an NVMe Controller when using the in-band tunneling mechanism. The entity sending the Request Message is collectively referred to as the Requester and the entity receiving the Request Message is collectively referred to as the Responder. After receiving a Request Message, the Responder processes the Request Message. When processing is complete, the Responder sends a Response Message back to the Requester.

A Request Message may be classified as a Command Message or a Control Primitive. Command Messages specify an operation to be performed by the Responder and may be further classified as an NVMe-MI Command, an NVMe Admin Command, or a PCIe Command. Control Primitives are used in the out-of-band mechanism to affect the servicing of a previously issued Command Message or get the state of a Command Slot and Management Endpoint (refer to section 4.2.1).

A Response Message may be classified as a Success Response or an Error Response.

Figure 24: NVMe-MI Message Taxonomy



4.1.1 Request Messages

Request Messages are NVMe-MI Messages that are generated by a Requester to send to a Responder.

Request Messages specify an action to be performed by the Responder. Request Messages are either Control Primitives (refer to section 4.2.1) or Command Messages. The format of the Message Body for a Command Message is command set specific and is specified by the NMIMT field in the Message Header.

The NVMe Management Interface supports three command sets:

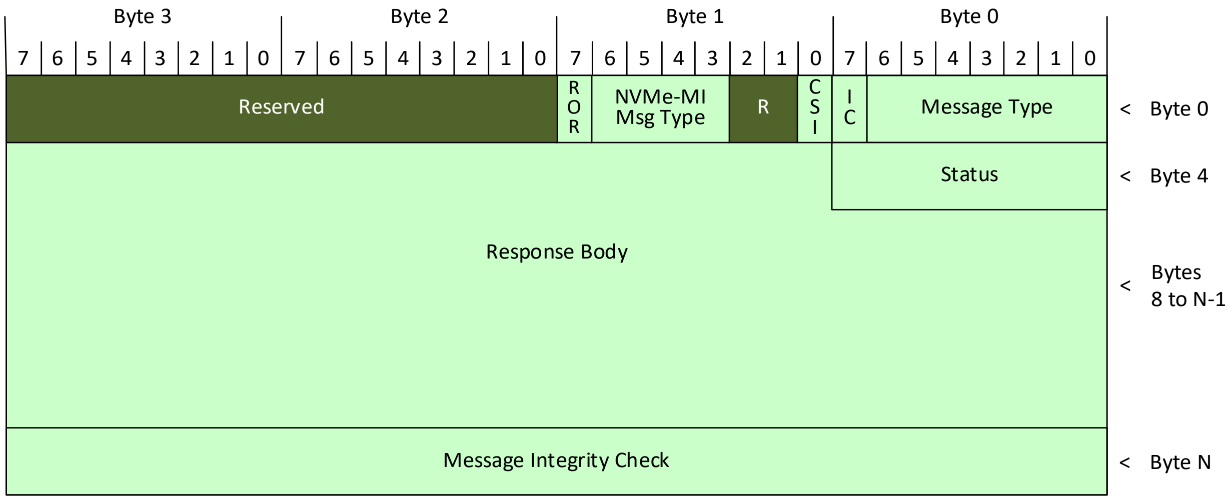
- The Management Interface Command Set is described in section 5;
- The NVM Express Admin Command Set is described in section 6; and
- The PCIe Command Set is described in section 7.

4.1.2 Response Messages

Response Messages are NVMe-MI Messages that are generated when a Responder has processed a previously issued Request Message.

The format of a Response Message is shown in Figure 25 and Figure 26. The first dword contains the Message Header. The Status field encodes the status associated with the Response Message. This is followed by the Response Body whose format is NVMe-MI Message Type and Response Message Status specific. Finally, if the Integrity Check (IC) bit is set to ‘1’, then the Response Message ends with the NVMe-MI Message Integrity Check field.

Figure 25: Response Message Format



In the out-of-band mechanism, the CSI bit in the Message Header specifies the Command Slot of the Request Message with which the Response Message is associated. In the in-band tunneling mechanism, the CSI bit in the Message Header is reserved.

The NVMe-MI Message Type (NMIMT) field contains the value from the same field in the corresponding Request Message.

Figure 26: Response Message Fields

Bytes	Description
3:0	NVMe-MI Message Header (NMH): Refer to section 3.1.
4	**Status (STATUS):** This field indicates the status associated with the Response Message. Response Message Status values are summarized in Figure 27.

Figure 26: Response Message Fields

Bytes	Description
N-1:5	**Response Body (RESPB):** This field contains response specific fields whose format is dependent on the NVMe-MI Message Type and Status field.
N+3:N	**Message Integrity Check:** Refer to section 3.1.

Response Message Status values are summarized in Figure 27. A Response Message Status of Success indicates that the corresponding Request Message completed successfully and that the Response Message is a Success Response. The format of the Response Body for a Success Response is dependent on the NVMe-MI Message Type and is described later in this specification.

A Response Message Status other than Success indicates that an error occurred during servicing of the corresponding Request Message and that the Response Message is an Error Response. The format of the Response Body is dependent on the Response Message Status. Figure 27 references the section that defines the format of the Response Message for each Response Message Status value. If multiple errors are present, a Responder may choose which error status to report.

Figure 27: Response Message Status Values

Value	Description	Response Message Format Section
00h	**Success:** The command completed successfully.	4.1.2.1
01h	**More Processing Required:** The Command Message is in progress and requires more time to complete processing. When this Response Message Status is used in a Response Message, a subsequent Response Message contains the result of the Command Message. This Response Message Status shall not be sent more than once per Command Message, except for retransmission due to a Replay Control Primitive as described in section 4.2.1.5.	4.1.2.3
02h	**Internal Error:** The Request Message could not be processed due to a vendor specific internal error.	4.1.2.1
03h	**Invalid Command Opcode:** The associated command opcode field is not valid. Invalid opcodes include reserved and optional opcodes that are not implemented.	4.1.2.1
04h	**Invalid Parameter:** Invalid parameter field value. Request Messages received with reserved or unimplemented values in defined fields shall be completed with an Invalid Parameter Error Response. Other error conditions that result in Invalid Parameter Error Response are specified elsewhere in this specification.	4.1.2.2
05h	**Invalid Command Size:** The size of the Message Body of the Request Message was different than expected due to a reason other than too much or too little Request Data (e.g., the Request Message did not contain all the required parameters or Request Data was present when not expected). The expected size of the Message Body is determined by the NVMe-MI Message Type and opcode assuming no other errors are detected (e.g., Invalid Command Opcode or Invalid Parameter).	4.1.2.1
06h	**Invalid Command Input Data Size:** The Command Message requires Request Data and contains too much or too little Request Data.	4.1.2.1
07h	**Access Denied:** A Request Message was prohibited from being processed due to a vendor specific protection mechanism or the Command and Feature Lockdown feature (refer to the NVM Express Base Specification).	4.1.2.1
08h to 1Fh	Reserved	-
20h	**VPD Updates Exceeded:** More updates to the VPD are attempted than allowed.	4.1.2.1
21h	**PCIe Inaccessible:** The PCIe functionality is not available at this time.	4.1.2.1

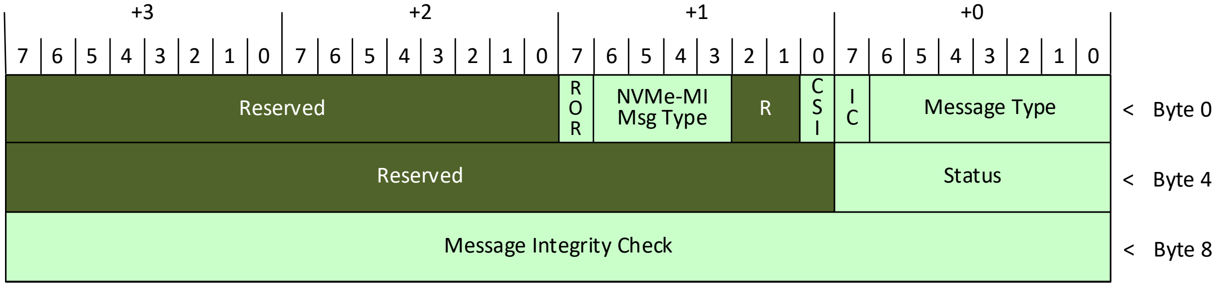
Figure 27: Response Message Status Values

Value	Description	Response Message Format Section
22h	**Management Endpoint Buffer Cleared Due to Sanitize:** An attempt was made to read data as defined in section 4.2.3 in the Management Endpoint Buffer that was zeroed due to a sanitize operation.	4.1.2.1
23h	**Enclosure Services Failure:** The Enclosure Services Process has failed in an unknown manner.	4.1.2.1
24h	**Enclosure Services Transfer Failure:** Communication with the Enclosure Services Process has failed.	4.1.2.1
25h	**Enclosure Failure:** An unrecoverable enclosure failure has been detected by the Enclosure Services Process.	4.1.2.1
26h	**Enclosure Services Transfer Refused:** The NVM Subsystem or Enclosure Services Process indicated an error or an invalid format in communication.	4.1.2.1
27h	**Unsupported Enclosure Function:** An SES Send command has been attempted to a simple Subenclosure.	4.1.2.1
28h	**Enclosure Services Unavailable:** The NVM Subsystem or Enclosure Services Process has encountered an error but may become available again.	4.1.2.1
29h	**Enclosure Degraded:** A noncritical failure has been detected by the Enclosure Services Process.	4.1.2.1
2Ah	**Sanitize In Progress:** The requested command is prohibited while a sanitize operation is in progress. Refer to section 8.1.	4.1.2.1
2Bh to DFh	Reserved	-
E0h to FFh	Vendor Specific	Vendor Specific

4.1.2.1 Generic Error Response

A Generic Error Response is generated for errors in which no additional information is provided beyond the Response Message Status. Bytes 7:5 are reserved. The format of a Generic Error Response is shown in Figure 28.

Figure 28: Generic Error Response



4.1.2.2 Invalid Parameter Error Response

An Invalid Parameter Error Response is generated for Error Responses where the Status field is set to Invalid Parameter. The format of an Invalid Parameter Error Response is shown in Figure 29 and the response specific fields are summarized in Figure 30.

Unless otherwise specified, if multiple invalid parameters errors exist in a Request Message, then the Management Endpoint selects the invalid parameter that is returned in the Invalid Parameter Error Response.

Figure 29: Invalid Parameter Error Response

<table>
<tr><td colspan="8">+3</td><td colspan="8">+2</td><td colspan="8">+1</td><td colspan="8">+0</td><td></td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td></td></tr>
<tr><td colspan="16">Reserved</td><td>ROR</td><td colspan="4">NVMe-MI Msg Type</td><td colspan="2">R</td><td>CSI</td><td>IC</td><td colspan="7">Message Type</td><td>< Byte 0</td></tr>
<tr><td colspan="24">Parameter Error Location (PEL)</td><td colspan="8">Status</td><td>< Byte 4</td></tr>
<tr><td colspan="32">Message Integrity Check</td><td>< Byte 8</td></tr>
</table>

Figure 30: Invalid Parameter Error Response Fields

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>7:5</td><td><b>Parameter Error Location (PEL):</b> This field indicates the request parameter within the Request Message that contains an invalid parameter.
<table><tr><th>Bits</th><th>Description</th></tr><tr><td>23:08</td><td>Least-significant byte of the Request Message of the parameter that contained the error. If the error is beyond byte 65,535, then the value 65,535 is reported in this field.</td></tr><tr><td>07:03</td><td>Reserved</td></tr><tr><td>02:00</td><td>Least-significant bit in the least-significant byte of the Request Message of the parameter that contained the error. Valid values are 0 to 7.</td></tr></table></td></tr>
</table>

4.1.2.3 More Processing Required Response

A More Processing Required Response shall be returned when the Management Endpoint requires more time to complete the processing of the Command Message as described in section 4.2.2.1. The format of a More Processing Required Response is shown in Figure 31 and the response specific fields are summarized in Figure 32.

Figure 31: More Processing Required Response

<table>
<tr><td colspan="8">+3</td><td colspan="8">+2</td><td colspan="8">+1</td><td colspan="8">+0</td><td></td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td></td></tr>
<tr><td colspan="16">Reserved</td><td>ROR</td><td colspan="4">NVMe-MI Msg Type</td><td colspan="2">R</td><td>CSI</td><td>IC</td><td colspan="7">Message Type</td><td>< Byte 0</td></tr>
<tr><td colspan="16">More Processing Required Time (MPRT)</td><td colspan="8">Reserved</td><td colspan="8">Status</td><td>< Byte 4</td></tr>
<tr><td colspan="32">Message Integrity Check</td><td>< Byte 8</td></tr>
</table>

Figure 32: More Processing Required Response Fields

Byte	Description
7:6	**More Processing Required Time (MPRT):** This field indicates the worst-case amount of additional time in 100 ms units that the Management Controller should wait for the Management Endpoint to complete processing of the Command Message. A value of FFFFh in this field indicates that greater than or equal to 6,553.5 s more processing time is required.

4.2 Out-of-Band Message Servicing Model

The out-of-band mechanism in this specification utilizes a request and response servicing model. A Management Controller sends a Request Message to a Management Endpoint, the Management Endpoint processes the Request Message, and when processing has completed, sends a Response Message back the Management Controller. Under no circumstances does a Management Endpoint generate an unsolicited Response Message (i.e., a Response Message that does not correspond to a previously received Request Message).

Unlike other NVMe-MI Messages that may span multiple MCTP packets, NVMe-MI Messages containing a Control Primitive shall consist of exactly one MCTP packet.

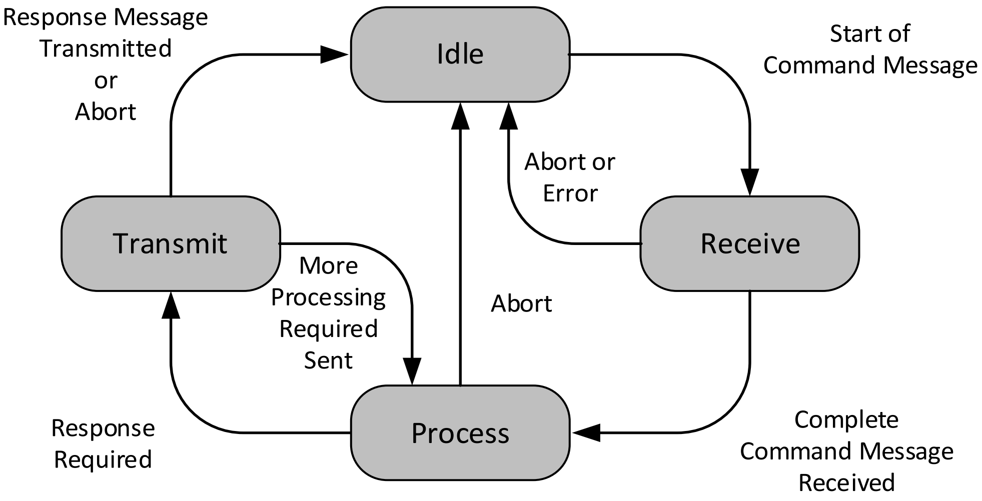
This specification utilizes Command Slots for Command Message servicing. A Management Controller should not send a new Command Message to a Command Slot until the Response Message for the previously issued Command Message to that Command Slot has been received. Each Management Endpoint contains two Command Slots that each include state information and a Pause flag (refer to section 4.2.1.4).

A Management Controller sends a Command Message to a Management Endpoint that targets a specific Command Slot in the Management Endpoint. The Management Endpoint assembles MCTP packets into Command Messages separately for each Command Slot. Each Command Slot remains allocated to the Command Message until servicing of the Command Message has completed.

A Command Message is the only type of multi-packet NVMe-MI Message that may be received by a Management Endpoint. The maximum number of Command Messages in flight to a Management Endpoint is equal to the number of Command Slots. The operation of each Command Slot is independent, allowing a Management Controller to have two independent streams of Command Messages to a Management Endpoint. The Command Message associated with each Command Slot is serviced in parallel. If the NVM Subsystem implements multiple Management Endpoints, then command servicing of each Management Endpoint occurs in parallel. An NVM Subsystem that implements *N* Management Endpoints may have up to 2*N* Command Messages serviced in parallel using the out-of-band mechanism.

The Command Servicing State Diagram in Figure 33 is used to describe functional requirements and does not mandate an implementation.

Figure 33: Command Servicing State Diagram



1. **Idle:** This is the default state of the command servicing state machine (e.g., following a reset). Command servicing transitions from Idle to the Receive state when the first MCTP packet of a Command Message is received (i.e., an MCTP packet with the SOM bit in the MCTP packet header set to '1' and the Message Type set to 4h).
2. **Receive:** The state when the first packet of a Command Message has been received and the Command Message is being assembled or validated. Command servicing transitions from Receive to the Idle state when an Abort Control Primitive is received, an error is detected in message assembly (refer to section 3.2.1.1), or the Message Integrity Check fails (refer to section 3.1.1.1). Command servicing transitions from Receive state to the Process state when a Command Message is assembled and the message integrity check is successful.
3. **Process:** The state when a Command Message is processed. Processing of a Command Message consists of checking for errors with the Command Message and performing the actions specified by the Command Message or aborting the Command Message. Command servicing transitions from Process to the Transmit state when a Response Message is required to be sent (i.e., processing of the Command Message has completed or command processing is expected to exceed the corresponding MCTP transport binding specification response timeout). Command servicing transitions from the Process state to the Idle state due to an Abort Control Primitive (r efer to section 4.2.1.3).
4. **Transmit:** The state in which a Response Message for the Command Message is transmitted to the Management Controller. Command servicing transitions from the Transmit to the Idle state once the entire NVMe-MI Message associated with the response to the Command Message has been transmitted on the physical medium or due to an Abort Control Primitive (refer to section 4.2.1.3).

   If, and only if, both the command servicing did not complete in the Process state and the Command Slot is not paused, then the Management Endpoint transmits a Response Message with status More Processing Required. If the Command Message requires more processing, then the Command Slot shall transition back to the Process state.

The behavior of receiving two or more overlapping Command Messages to the same Command Slot is undefined. If this results in the Management Endpoint discarding a Command Message, then this is considered receiving a Command Message to a non-Idle Command Slot (CMNICS). Refer to section 4.2.1.4.

4.2.1 Control Primitives

Control Primitives are Request Messages sent from a Management Controller to a Management Endpoint to affect the servicing of a previously issued Command Message or get the state of a Command Slot and Management Endpoint. Control Primitives are applicable only in the out-of-band mechanism and are prohibited in the in-band tunneling mechanism.

Control Primitives may target a Command Slot. Unlike Command Messages, Control Primitives may be sent while the Command Slot is in any command servicing state and are processed immediately by the Management Endpoint. Unless otherwise indicated, Control Primitives do not change the command servicing state of the Command Slot.

The format of a Control Primitive is shown in Figure 34 and the fields are described in Figure 35.

Figure 34: Control Primitive Request Message Format

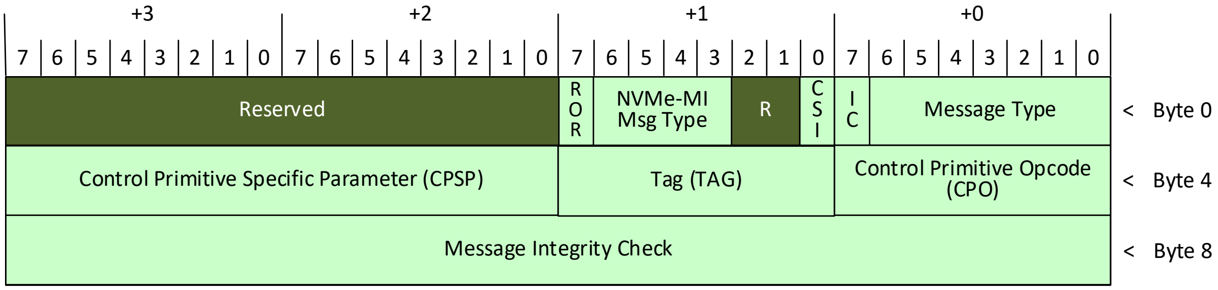


Figure 35: Control Primitive Fields

Bytes	Description
03:00	**NVMe-MI Message Header (NMH):** Refer to section 3.1.
04	**Control Primitive Opcode (CPO):** This field specifies the opcode of the Control Primitive to be processed. Refer to Figure 36.
05	**Tag (TAG):** This field contains an opaque value that is sent from the Management Controller in the Control Primitive and returned by the Management Endpoint in the associated Response Message. A Management Controller is allowed to use any value in this field.
07:06	**Control Primitive Specific Parameter (CPSP):** This field is used to pass Control Primitive specific parameter information.
11:08	**Message Integrity Check (MIC):** Refer to section 3.1.

Figure 36: Control Primitive Opcodes

Opcode	O/M[1]	Command
00h	M	Pause
01h	M	Resume
02h	M	Abort
03h	M	Get State
04h	M	Replay
05h to EFh		Reserved
F0h to FFh	O	Vendor Specific
NOTES: 1. O/M: O = Optional, M = Mandatory.		

The format of a Success Response associated with a Control Primitive is shown in Figure 37 and the fields are described in Figure 38.

Figure 37: Control Primitive Success Response Format

<table>
<tr><td colspan="8">+3</td><td colspan="8">+2</td><td colspan="8">+1</td><td colspan="8">+0</td><td></td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td></td></tr>
<tr><td colspan="16">Reserved</td><td>ROR</td><td colspan="4">NVMe-MI Msg Type</td><td colspan="2">R</td><td>CSI</td><td>IC</td><td colspan="7">Message Type</td><td>< Byte 0</td></tr>
<tr><td colspan="16">Control Primitive Specific Response (CPSR)</td><td colspan="8">Tag (TAG)</td><td colspan="8">Status</td><td>< Byte 4</td></tr>
<tr><td colspan="32">Message Integrity Check</td><td>< Byte 8</td></tr>
</table>

Figure 38: Control Primitive Success Response Fields

Bytes	Description
03:00	**NVMe-MI Message Header (NMH):** Refer to section 3.1.
04	**Status (STATUS):** Refer to section 4.1.2.
05	**Tag (TAG):** This field contains an opaque value that is passed by the Management Endpoint from the Control Primitive to the associated Response Message. The Response Message contains the same value in this field as the corresponding Request Message.
07:06	**Control Primitive Specific Response (CPSR):** This field is used to return Control Primitive specific status.
11:08	**Message Integrity Check (MIC):** Refer to section 3.1.

A Management Endpoint transmits a Response Message to the Management Controller when the actions associated with that Control Primitive have completed.

Unlike Command Messages, a Management Controller may issue a Control Primitive to a Command Slot without waiting for a response for previously issued Control Primitives to that Command Slot. If multiple Control Primitives are issued without waiting for responses from the Management Endpoint, only the actions and response associated with the last Control Primitive are guaranteed (i.e., the actions associated with previously issued but unacknowledged Control Primitives may or may not be performed and the Response Messages for previously issued but unacknowledged Control Primitives may or may not be transmitted). Receipt of a Control Primitive never corrupts a previous Control Primitive associated with the Command Slot. The Response Message is either entirely transmitted or discarded.

The TAG field is an opaque value copied from the Control Primitive Request Message into the Response Message. By using unique TAG values, it is possible for the Management Controller to link Response Messages with Request Messages.

4.2.1.1 Pause

The Pause Control Primitive is used to suspend response transmission and suspend the timeout waiting for packet for both Command Slots in a Management Endpoint. The CSI bit in a Pause Control Primitive is not used and shall be cleared to 0h. If the CSI bit is set to '1', then the Management Endpoint should transmit an Invalid Parameter Error Response with the PEL field indicating the CSI bit.

Associated with each Command Slot is a Pause Flag that determines whether the slot is 'paused.' The Pause Flag status is included with a Success Response and may also be read using the Get State Control Primitive.

The CPSP field for the Pause Control Primitive is reserved.

The format of the CPSR field in the Control Primitive Success Response is shown in Figure 39.

Figure 39: Pause Control Primitive Success Response Fields

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>07:06</td><td><b>Control Primitive Specific Response (CPSR):</b> This field is used to return Control Primitive specific status.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>15:02</td><td>Reserved</td></tr>
<tr><td>01</td><td><b>Pause Flag Status Slot 1 (PFSS1):</b> This bit indicates whether or not Command Slot 1 is paused after completing the Pause Control Primitive. This bit set to ‘1’ indicates the Command Slot is paused. This bit cleared to ‘0’ indicates the Command Slot is not paused.</td></tr>
<tr><td>00</td><td><b>Pause Flag Status Slot 0 (PFSS0):</b> This bit indicates whether or not Command Slot 0 is paused after completing the Pause Control Primitive. This bit set to ‘1’ indicates the Command Slot is paused. This bit cleared to ‘0’ indicates the Command Slot is not paused.</td></tr>
</table>
</td></tr>
</table>

The result of a Pause Control Primitive on a Command Slot is dependent on the command servicing state of the Command Slot when the Pause Control Primitive is received, as described below:

Idle: The Pause Control Primitive has no effect, and the Pause Flag is not changed (i.e., remains cleared to ‘0’). Refer to section 4.2.1.4.

Receive: The Pause Control Primitive sets the Pause Flag to ‘1’ (refer to section 4.2.1.4) and alerts the Management Endpoint that remaining MCTP packets associated with the Command Message may be delayed. Further packets sent to this Command Slot while the Pause Flag is set to ‘1’ are received normally.

Process: The Pause Control Primitive sets the Pause Flag to ‘1’ (refer to section 4.2.1.4). The Pause Flag has no effect on the command processing in the Command Slot. Upon completion of command processing, the Command Slot shall transition to the Transmit state.

Transmit: The Pause Control Primitive sets the Pause Flag to ‘1’ (refer to section 4.2.1.4) suspending transmission of Response Messages on a packet boundary. The Management Endpoint should pause transmission as soon as possible after receiving a Pause Control Primitive.

The Management Endpoint shall transmit a Response Message with success status after receiving the Pause Control Primitive. It is not an error to issue a Pause Control Primitive when a Command Slot is already paused.

While the Pause Flag is set to ‘1’, the Management Endpoint disables the timeout waiting for packet timer and does not transmit Response Messages to Command Messages. The timeout waiting for a packet is the lesser of 100 ms or the time defined in the appropriate MCTP transport binding specification. The Management Controller should not send Command Messages to a Command Slot that is paused.

4.2.1.2 Resume

The Resume Control Primitive is used to resume from a paused condition. This is the complement to the Pause Control Primitive.

Like the Pause Control Primitive, the Resume Control Primitive affects both slots and the CSI bit in a Resume Control Primitive shall be cleared to ‘0’. If a Command Slot was not paused before receiving the Resume Control Primitive, the Resume Control Primitive completes successfully and has no effect.

The CPSP field for the Resume Control Primitive is reserved. The CPSR field in the Control Primitive Success Response is reserved.

The result of a Resume Control Primitive is based on the state of a Command Slot when the Resume Control Primitive is received, as described below:

Idle: The Resume Control Primitive has no effect.

Receive: The Resume Control Primitive alerts the Management Endpoint that transmission of any remaining MCTP packets associated with the Command Message is resuming. The Pause Flag is cleared to '0' (refer to section 4.2.1.4).

Process: If the Command Slot is paused and a More Processing Required Response has not yet been transmitted for the Command Message being processed, then the request-to-response timer shall be reset and restarted (refer to section 4.2.2.1 for details on the request-to-response time). The Pause Flag is cleared to '0' (refer to section 4.2.1.4).

Transmit: The Management Endpoint resumes transmission of the Response Message corresponding to the Command Message associated with the Command Slot after responding to the Resume Control Primitive. The Pause Flag is cleared to '0' (refer to section 4.2.1.4).

The Management Endpoint shall transmit a Control Primitive Response Message with success status after receiving the Resume Control Primitive.

4.2.1.3 Abort

The Abort Control Primitive is used to re-initialize a Command Slot to the Idle state, clear the Pause Flag associated with that Command Slot to '0', and attempt to abort command servicing associated with that Command Slot.

Aborting a Command Message shall have no effect on the other Command Slot of the Management Endpoint, other Management Endpoints, or NVMe Controllers in the NVM Subsystem. Subsequent command servicing in the Command Slot is not affected by the Abort Control Primitive.

A Management Controller may issue an Abort Control Primitive to clean-up resources associated with a Command Slot in an unknown state.

The CPSP field for the Abort Control Primitive is reserved. The format of the CPSR field in the Control Primitive Success Response is shown in Figure 40.

Figure 40: Abort Control Primitive Success Response Fields

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>07:06</td><td><b>Control Primitive Specific Response (CPSR):</b> This field is used to return Control Primitive specific status.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>15:02</td><td>Reserved</td></tr>
<tr><td>01:00</td><td><b>Command Processing Abort Status (CPAS):</b> This field indicates the effect of the Abort Control Primitive on the processing of the Command Message associated with the Command Slot.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>00b</td><td>Command aborted after processing completed or no command to abort.</td></tr>
<tr><td>01b</td><td>Command aborted before processing began.</td></tr>
<tr><td>10b</td><td>Command processing partially completed.</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

The result of an Abort Control Primitive is based on the command servicing state of the specified Command Slot when the Abort Control Primitive is received, as described below:

Idle: The Abort Control Primitive has no effect. The Management Endpoint shall transmit a Response Message with success status and the CPAS field cleared to 0h.

Receive: The Management Endpoint discards the contents of the Command Slot and transitions to the Idle state. The Management Endpoint shall transmit a Response Message with success status and the CPAS field set to 1h.

Process: The Abort Control Primitive causes processing of the command in the Command Slot to be aborted:

a) If the Abort Control Primitive was received before command processing started, the Management Endpoint discards the contents of the Command Slot and transitions to the Idle state. The Management Endpoint shall transmit a Success Response and the CPAS field set to 1h; or
b) If the Abort Control Primitive was received while the command is being processed, the Management Endpoint discards the contents of the Command Slot and transitions to the Idle state. The Management Endpoint attempts to abort the command:

    - If the command is aborted and had no effect on the NVM Subsystem, then the Management Endpoint shall transmit a Success Response and the CPAS field set to 1h; or
    - If the Management Endpoint is not able to abort the command, then the Management Endpoint shall transmit a Success Response and set the CPAS field to 2h.

Transmit: The Management Endpoint discards the contents of the Command Slot and transitions to the Idle state. The Management Endpoint transmits a Response Message with success status and the CPAS field cleared to 0h.

It is not a Management Endpoint error if the Management Controller issues an Abort Control Primitive to a Command Slot that is paused. The state of Command Slot is reinitialized clearing the Pause Flag to ‘0’.

4.2.1.4 Get State

The Get State Control Primitive is used to get the state of a Command Slot and Management Endpoint.

The format of the CPSP field in the Control Primitive Request Message is shown in Figure 41.

Figure 41: Get State Control Primitive Request Message Fields

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>07:06</td><td><b>Control Primitive Specific Parameter (CPSP):</b> This field is used to pass Control Primitive specific parameter information.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>15:01</td><td>Reserved</td></tr>
<tr><td>00</td><td><b>Clear Error State Flags (CESF):</b> This bit specifies whether or not to clear the error state flags when completing this command.</td></tr>
</table>
</td></tr>
</table>

The Management Endpoint shall transmit a Response Message with success status after receiving the Get State Control Primitive. The format of the CPSR field in the Control Primitive Success Response is shown in Figure 42.

Figure 42: Get State Control Primitive Success Response Fields

Bytes	Description
07:06	**Control Primitive Specific Response (CPSR):** This field is used to return Control Primitive specific status.

Bits	Command Slot Specific[1]	Description
15	Yes	**Pause Flag (PFLG):** This bit indicates whether or not the Command Slot is paused. This bit set to '1' indicates the Command Slot is paused. This bit cleared to '0' indicates the Command Slot is not paused. While the Pause Flag is set, the Management Endpoint disables the timeout waiting for packet timer (refer to section 4.2.1.1) for the Command Slot and does not transmit responses to Command Messages.
14	No	**NVM Subsystem Reset Occurred (NSSRO):** This bit indicates when an NVM Subsystem Reset occurs while main power is applied. This bit is set to '1' if the last occurrence of an NVM Subsystem Reset occurred while main power was applied to the NVM Subsystem. This bit is cleared to '0' following a power cycle and following a Get State Control Primitive with the CESF bit set to '1'.
13	No	**Bad Packet or Other Physical Layer (BPOPL):** This bit is set to '1' if a packet sent to the Management Endpoint failed a transport specific packet integrity check since the last time Get State Control Primitive was processed with the CESF bit set to '1'.
12	No	**Bad, Unexpected, or Expired Message Tag (BUEMT):** This bit is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State Control Primitive was processed with the CESF bit set to '1'.
11	No	**Out-of-Sequence Packet Sequence Number (OSPSN):** This bit is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State Control Primitive was processed with the CESF bit set to '1'.
10	No	**Unexpected Middle or End of Packet (UMEP):** This bit is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State Control Primitive was processed with the CESF bit set to '1'.
09	No	**Incorrect Transmission Unit (ITU):** This bit is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State Control Primitive was processed with the CESF bit set to '1'.
08	No	**Unknown Destination ID (UDSTID):** This bit is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State Control Primitive was processed with the CESF bit set to '1'.
07	No	**Bad Header Version (BHVS):** This bit is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State Control Primitive was processed with the CESF bit set to '1'.
06	No	**Unsupported Transmission Unit (UTUNT):** This bit is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State Control Primitive was processed with the CESF bit set to '1'.
05	No	**Timeout Waiting for a Packet (WPTT):** This bit is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State Control Primitive was processed with the CESF bit set to '1'.
04	No	**Bad Message Integrity Check Error (BMICE):** This bit is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State Control Primitive was processed with the CESF bit set to '1'.
03	No	**Command Message to non-Idle Command Slot (CMNICS):** This bit is set to '1' if the Management Endpoint discarded one or more Command Messages due to overlapping Command Messages to a Command Slot since the last time Get State Control Primitive was processed with the CESF bit set to '1'.
02		Reserved

Figure 42: Get State Control Primitive Success Response Fields

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>01:00</td><td>Yes</td><td><b>Slot Command Servicing State (SSTA):</b> This field indicates the current command servicing state of the Command Slot. An implementation may choose to indicate only the Idle and Process states in this field. Refer to Figure 33.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>Idle</td></tr>
<tr><td>1h</td><td>Receive</td></tr>
<tr><td>2h</td><td>Process</td></tr>
<tr><td>3h</td><td>Transmit</td></tr>
</table>
</td></tr>
<tr><td colspan="3">NOTES:
1. Command Slot Specific. A ‘Yes’ in this column indicates the value of the field is independent per Command Slot within a Management Endpoint. A ‘No’ in this column indicates the same value is reported for either Command Slot.</td></tr>
</table>
</td></tr>
</table>

4.2.1.5 Replay

The Replay Control Primitive is used to retransmit the Response Message for the last Command Message processed in a Command Slot. The replayed Response Message forms a new MCTP Response Message with Message Data starting from Response Replay Offset of the original Response Message and continuing to the end of the Response Message, including the original MIC. The first packet shall have SOM set to ’1’ and shall include the Message Header of the original Response Message even if the Response Replay Offset is not 0h. The Msg tag in each packet of the replayed Response Message shall be set to the value of the Msg tag in the associated Replay Control Primitive. Refer to the MCTP Base Specification for the definition of the Msg tag.

Note that the Management Controller requires extensions to the MCTP Base Specification in its MCTP layer in order to replay a Response Message using a non-zero Response Replay Offset. No extensions to the MCTP Base Specification are required to replay with Response Replay Offset equal to 0h. For the case where a Management Controller chooses to use a non-zero Response Replay Offset, the MCTP Base Specification requires terminating message assembly for certain errors (i.e., receiving a packet with bad packet data integrity).

If a Management Controller receives a number of packets with no errors in a Response Message and then gets an error on a packet that causes termination of message assembly, the Management Controller requires extensions in its MCTP layer to forward the packets it received with no errors to its NVMe-MI layer prior to terminating message assembly. The Management Controller can then issue a replay to get the second part of the Response Message using a non-zero Response Replay Offset. The Management Controller’s NVMe-MI layer then assembles the two partial Response Messages to create the whole Response Message. The MIC can then be validated across the whole Response Message as described in section 3.1.1.1.

The format of the CPSP field in the Control Primitive Request Message is shown in Figure 43.

Figure 43: Replay Control Primitive Request Fields

<table>
<tr><th>Bytes</th><th colspan="2">Description</th></tr>
<tr><td rowspan="4">07:06</td><td colspan="2">**Control Primitive Specific Parameter (CPSP):** This field is used to pass Control Primitive specific parameter information.</td></tr>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>15:08</td><td>Reserved</td></tr>
<tr><td>07:00</td><td>**Response Replay Offset (RRO):** This field specifies the starting packet number from which the Response Message associated with the last Command Message processed in the Command Slot shall be replayed.
This is a 0's based value. When this field is cleared to 0h, the first packet of the associated Response Message is the first packet replayed.
If this field specifies an offset that is beyond the length of the Response Message, then processing of the Control Primitive is aborted and the Management Endpoint shall transmit an Invalid Parameter Error Response with the PEL field indicating this field.</td></tr>
</table>

The format of the CPSR field in the Control Primitive Success Response is shown in Figure 44.

Figure 44: Replay Control Primitive Success Response Fields

<table>
<tr><th>Bytes</th><th colspan="2">Description</th></tr>
<tr><td rowspan="4">07:06</td><td colspan="2">**Control Primitive Specific Response (CPSR):** This field is used to return Control Primitive specific status.</td></tr>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>15:01</td><td>Reserved</td></tr>
<tr><td>00</td><td>**Response Replay (RR):** This bit indicates if a previous Response Message is retransmitted. This bit is set to '1' if the requested Response Message is retransmitted by the Management Endpoint. This bit is cleared to '0' if the requested Response Message is not retransmitted.</td></tr>
</table>

The result of a Replay Control Primitive is based on the command servicing state of the specified Command Slot when the Replay Control Primitive is received, as described below:

Idle: The Replay Control Primitive requests retransmission of the completion at the offset specified by the RRO field if such a completion is available:

a) If the Replay Control Primitive was received following an Abort Control Primitive or a reset (refer to section 8.3) before any Command Messages are processed, then there is no Response Message available to retransmit. The Management Endpoint shall transmit a Response Message with success status with the RR bit cleared to '0'; or
b) If the Replay Control Primitive was received following the processing of one or more Command Messages, then the Management Endpoint shall transmit a Response Message with success status with the RR bit set to '1'. The Management Endpoint transmits the MCTP packets associated with the requested Response Message after the Control Primitive Success Response.

Receive: The Management Endpoint transmits a Response Message with success status with the RR bit cleared to '0'.

Process: If a More Processing Required Response has not been transmitted for the Command Message being processed, then a Success Response shall be transmitted with the RR bit cleared to '0'.

If a More Processing Required Response has been transmitted, then a Success Response shall be transmitted with the RR bit set to '1' and then the More Processing Required Response shall be retransmitted. The Management Endpoint shall update the More Processing Required Time field in the Response Message with the current worst-case amount of additional time that the Management Controller should wait for the Management Endpoint to complete processing of the Command Message.

Transmit: The Management Endpoint stops transmitting response packets for the Command Slot and then transmits a Response Message with success status with the RR bit set to ‘1’. The Management Endpoint transmits a Response Message containing the packets starting at the packet offset specified in the Response Replay Offset field of the Replay Control Primitive after the Control Primitive Success Response. The Command Slot remains in the Transmit state until retransmission is complete.

It is not an error to issue a Replay Control Primitive to a Command Slot that is paused. A response is transmitted even if the Command Slot is paused at any time during the response, including before the first packet was transmitted. After successful completion of the Replay Control Primitive, neither Command Slot is paused (i.e., there is an implicit Resume Control Primitive affecting both Command Slots when processing the Replay Control Primitive).

4.2.2 Out-of-Band Error Handling

This section describes error handling specific to the NVMe-MI out-of-band message processing model.

4.2.2.1 Command Timeouts

MCTP defines a maximum response time for MCTP control messages (refer to the appropriate MCTP transport binding specification).

If a Management Endpoint determines that command processing may not complete in less than or equal to the lesser of 100 ms or the request-to-response time specified in the appropriate MCTP transport binding specification, the Management Endpoint shall return a More Processing Required Response as described in section 4.1.2.3. The More Processing Required Response from the Management Endpoint is allowed to be delayed beyond this timeout while the transport is busy or unavailable.

A Management Endpoint should only return a More Processing Required Response for Command Messages that are expected to take longer than the required time (e.g., Format NVM). Implementations are strongly discouraged from using this response while processing Commands Messages that take less than or equal to the lesser of 100 ms or the request-to-response time specified in the appropriate MCTP transport binding specification.

4.2.2.2 Control Primitive Timeouts

A Management Endpoint shall attempt to respond to a Control Primitive within the lesser of 100 ms or the request-to-response time specified in the appropriate MCTP transport binding specification. The Response Message from the Management Endpoint is allowed to be delayed beyond this timeout while the transport is busy or unavailable.

4.2.3 Management Endpoint Buffer

Since the maximum size of the NVMe-MI Message is 4,224 bytes, the maximum possible amount of out-of-band Request Data that may be contained in a Request Message is 4,216 bytes (i.e., 4,224 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 4,215 bytes (i.e., 4,224 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 additional command specific fields. In some cases, it is desirable to service Command Messages that contain more

Request Data or Response Data than allowed to 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. A Management Endpoint Buffer is exclusive to one Management Endpoint and shall not be shared. 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 are not required to 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 partial 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) bit is set to ‘1’ in a Command Message that normally contains Request Data, then Request Data shall not be transferred in the Command Message itself and the required Request Data shall be transferred from the Management Endpoint Buffer. The Request Data starts at a zero offset from the start of the Management Endpoint Buffer. If that Command Message contains Request Data or does not support Request Data, then the Management Endpoint responds with an Invalid Parameter Error Response with the PEL field indicating the Request Data field of the Command Message.

If the Management Endpoint Buffer (MEB) bit 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 cleared to 0h 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 with the MEB bit 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 cleared to 0h (i.e., the Message 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 (refer to the NVM Express Base Specification) performed in an NVM Subsystem. The MCTP Management Endpoint Buffer may contain Response Data associated with a previously processed command that is not allowed during a sanitize operation. When a sanitize operation is initiated, the contents of the Management Endpoint Buffer shall be cleared 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 Response Message Status of Management Endpoint Buffer Cleared Due to Sanitize. This Response Message Status 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.

4.3 In-Band Tunneling Message Servicing Model

The in-band tunneling mechanism in this specification utilizes two NVMe Admin Commands (NVMe-MI Send and NVMe-MI Receive). The NVMe-MI Send command is used to tunnel an NVMe-MI Command from host software to an NVMe Controller that transfers data from the host to the NVMe Controller (similar to a write operation) or to instruct the Responder to perform an action (e.g., to reset the NVM Subsystem using the Reset command). The NVMe-MI Receive command is used to tunnel an NVMe-MI Command from a host to an NVMe Controller that transfers data from the NVMe Controller to the host (similar to a read operation). Figure 60 specifies whether an NVMe-MI Command is tunneled via the NVMe-MI Send command or the NVMe-MI Receive command.

Refer to the NVM Express Base Specification for additional details on the NVMe-MI Send and NVMe-MI Receive commands. Additional details on NVMe-MI Send are in section 4.3.1 and additional details on NVMe-MI Receive are in section 4.3.2.

4.3.1 NVMe-MI Send Command

The NVMe-MI Send command is an NVMe Admin Command as defined by this specification and the NVM Express Base Specification. It is used to tunnel an NVMe-MI Command in-band from host software to an NVMe Controller that transfers data from a host to an NVMe Controller (similar to a write operation) or to instruct the Responder to perform an action (e.g., to reset the NVM Subsystem using the Reset command). The data being transferred or action to be performed is in one or more of the following locations: Request Data, NVMe Management Dword 0, NVMe Management Dword 1. Figure 60 specifies which NVMe-MI Commands are tunneled via the NVMe-MI Send command.

NVMe-MI Commands may apply to the NVM Subsystem, Controllers, and/or Namespaces. If the tunneled NVMe-MI Command applies to one or more Controllers, then the applicable Controller(s) are specified by fields in the tunneled NVMe-MI Command. Note that unlike some other Admin Commands, the Controller to which the NVMe-MI Send command is issued is not used to determine which Controller the tunneled NVMe-MI Command applies to. If the tunneled NVMe-MI Command applies to one or more Namespaces, then the applicable Namespace(s) are specified by fields in the tunneled NVMe-MI Command. Note that the Namespace Identifier (NSID) field of the NVMe-MI Send command (bytes 7:4 of the Submission Queue Entry) is not used and should be cleared to 0h by host software.

The mapping of how an NVMe-MI Command is tunneled inside of NVMe-MI Send commands is described in section 4.3.1.1. The NVMe-MI Send command servicing model is described in section 4.3.1.2.

4.3.1.1 NVMe-MI Send Command Request Message to NVMe Admin Command SQE Mapping

In order to tunnel an NVMe-MI Command in-band via NVMe-MI Send, an NVMe-MI Request Message is mapped onto an NVMe Submission Queue Entry (SQE) as shown pictorially in Figure 45 and in table form in Figure 46. An NVMe-MI Response Message is mapped on to an NVMe Completion Queue Entry (CQE) as shown pictorially in Figure 47 and in table form in Figure 48. Refer to the NVM Express Base Specification for details on an NVMe Submission Queue Entry and an NVMe Completion Queue Entry.

Figure 45: NVMe-MI Send Command Request Message to NVMe Admin Command SQE Mapping Diagram

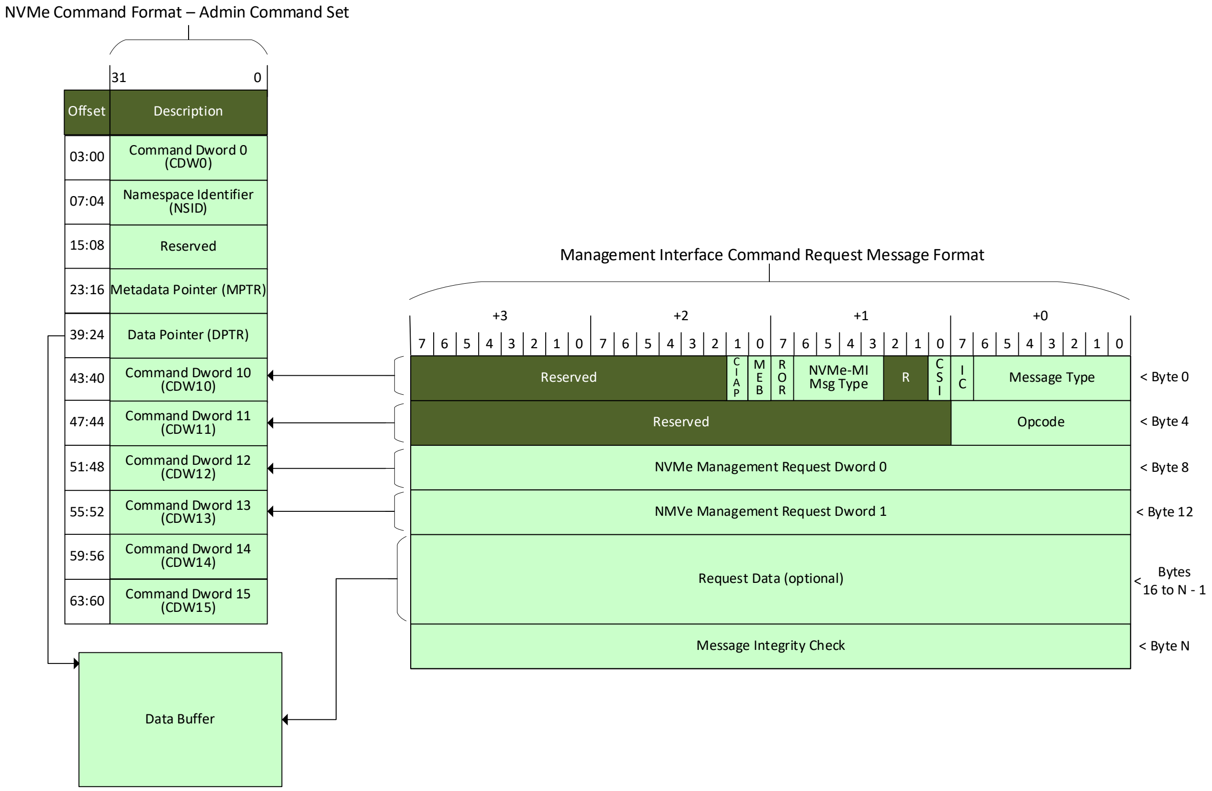


Figure 46: NVMe-MI Send Command Request Message to NVMe Admin Command SQE Mapping Table

NVMe-MI Command Request Message		NVMe Admin Command SQE Mapping	
Bytes	**Description**	**Bytes**	**Description**
Not applicable (n/a)	This field has no equivalent in this specification.	03:00	**Command Dword 0 (CDW0):** Refer to the NVM Express Base Specification.
n/a	If the tunneled NVMe-MI Command requires one or more Namespaces to be specified, then the applicable Namespace Identifiers are specified by the tunneled NVMe-MI Command.	07:04	**Namespace Identifier (NSID):** This field should be cleared to 0h by host software. Refer to the NVM Express Base Specification for more details.

Figure 46: NVMe-MI Send Command Request Message to NVMe Admin Command SQE Mapping Table

NVMe-MI Command Request Message		NVMe Admin Command SQE Mapping	
Bytes	**Description**	**Bytes**	**Description**
n/a	These bytes have no equivalent in this specification.	23:08	Refer to the NVM Express Base Specification.
n/a	There is no equivalent of DPTR in this specification. In NVMe-MI Send, the Request Data is included in the Request Data portion of the Request Message.	39:24	**Data Pointer (DPTR):** This field contains a pointer to the start of the data buffer that contains the Request Data portion of the NVMe-MI Command that is being tunneled. If there is no Request Data for this command, then this field is ignored. Refer to the NVM Express Base Specification for the definition of this field.
03:00	NVMe-MI Message Header (NMH)	43:40	**Command Dword 10 (CDW10):** Dword 0 of the Request Message (NMH) that is being tunneled maps to CDW10 of the SQE. The byte ordering within CDW10 is little endian (i.e., NMH byte 0 maps to CDW10 byte 0, NMH byte 1 maps to CDW10 byte 1, etc.).
04	Opcode (OPC)	47:44	**Command Dword 11 (CDW11):** Dword 1 of the Request Message (OPC and Reserved bytes 7:5) that is being tunneled maps to CDW11 of the SQE. The byte ordering within CDW11 is little endian (i.e., OPC maps to CDW11 byte 0, the LSB of the Reserved field (NVMe-MI Command Request Message byte 5) maps to CDW11 byte 1, etc.).
07:05	Reserved		
11:08	NVMe Management Dword 0 (NMD0)	51:48	**Command Dword 12 (CDW12):** Dword 2 of the Request Message (NMD0) that is being tunneled maps to CDW12 of the SQE. The byte ordering within CDW12 is little endian (i.e., NMD0 byte 0 maps to CDW12 byte 0, NMD0 byte 1 maps to CDW12 byte 1).
15:12	NVMe Management Dword 1 (NMD1)	55:52	**Command Dword 13 (CDW13):** Dword 3 of the Request Message (NMD1) that is being tunneled maps to CDW13 of the SQE. The byte ordering within CDW13 is little endian (i.e., NMD1 byte 0 maps to CDW13 byte 0, NMD1 byte 1 maps to CDW13 byte 1).
n/a	This field has no equivalent in this specification.	59:56	**Command Dword 14 (CDW14):** Reserved.
n/a	This field has no equivalent in this specification.	63:60	**Command Dword 15 (CDW15):** Reserved.
N-1:16	Request Data (REQD)	n/a	Request Data is placed by host software into the data buffer pointed to by DPTR. If the Request Data is not dword granular, then the Request Data shall be padded with the minimum number of bytes cleared to 0h to make the Request Data dword granular. The byte ordering within the data buffer pointed to by DPTR is little endian (i.e., REQD byte 0 maps to byte 0 of the data buffer pointed to by DPTR, REQD byte 1 maps to byte 1 of the data buffer pointed to by DPTR, etc.).
N+3:N	Message Integrity Check (MIC)	n/a	The Message Integrity Check is not used in the in-band tunneling mechanism.

Figure 47: NVMe-MI Send Command Response Message to NVMe Admin Command CQE Mapping Diagram

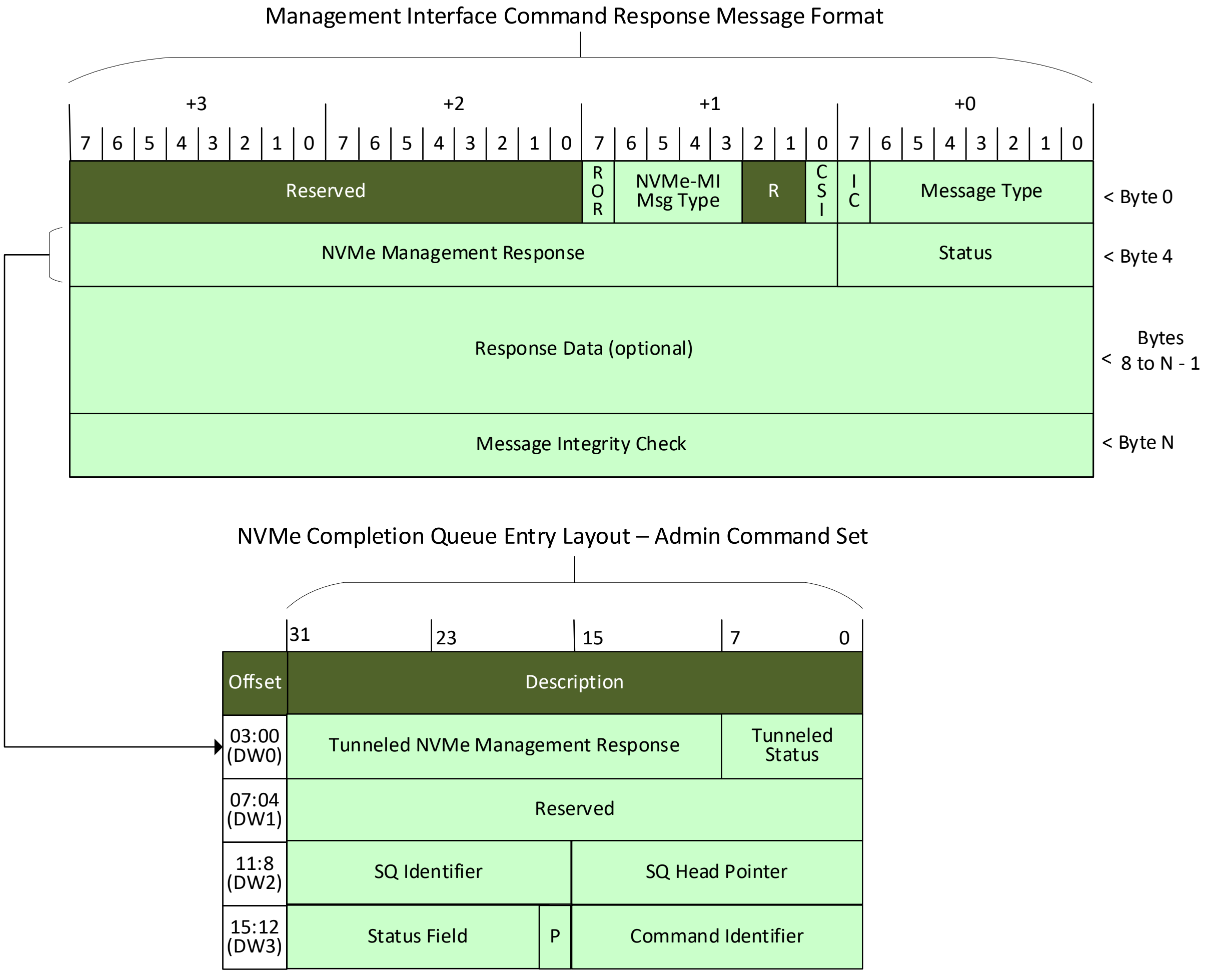


Figure 48: NVMe-MI Send Command Response Message to NVMe Admin Command CQE Mapping Table

NVMe-MI Command Response Message		NVMe Admin Command CQE Mapping	
Bytes	**Description**	**Bytes**	**Description**
00	MCTP Data (MCTPD)	n/a	This field has no equivalent in the NVMe Admin Command CQE.
01	NVMe-MI Message Parameters (NMP)	n/a	This field has no equivalent in the NVMe Admin Command CQE.
03:02	Reserved	n/a	This field has no equivalent in the NVMe Admin Command CQE.
04	Status (STATUS)	03:00	**Command Specific (DW0):** Dword 1 of the Response Message (STATUS and NMRESP) that is being tunneled maps to DW0 of the CQE. The byte ordering within DW0 is little endian (i.e., STATUS maps to DW0 byte 0, the LSB of the NMRESP field (NVMe-MI Command Response Message byte 5) maps to DW0 byte 1, etc.). Refer to Figure 49 for additional details on this field.
07:05	NVMe Management Response (NMRESP)		

Figure 48: NVMe-MI Send Command Response Message to NVMe Admin Command CQE Mapping Table

NVMe-MI Command Response Message		NVMe Admin Command CQE Mapping	
Bytes	**Description**	**Bytes**	**Description**
N-1:8	Response Data (RESPD)	n/a	There is no Response Data for NVMe-MI Send.
N+3:N	Message Integrity Check (MIC)	n/a	The Message Integrity Check is not used in the in-band tunneling mechanism.
n/a	These bytes have no equivalent in this specification.	15:04	Refer to the NVM Express Base Specification.

The definition of Dword 0 of the completion queue entry is in Figure 49.

Figure 49: NVMe-MI Send – Completion Queue Entry Dword 0 (NSCQED0)

Bytes	Description
31:08	**Tunneled NVMe Management Response (TNMRESP):** This field contains the NVMe Management Response field from the NVMe-MI Command that is being tunneled in-band. If any errors are detected in the NVMe Context as described in section 4.3.1.2, then this field shall be cleared to 0h.
07:00	**Tunneled Status (TSTAT):** This field contains the Status field from the NVMe-MI Command that is being tunneled in-band. If any errors are detected in the NVMe Context as described in section 4.3.1.2, then this field shall be cleared to 0h.

4.3.1.2 NVMe-MI Send Command Servicing Model

The NVMe-MI Send command servicing model is illustrated in Figure 50 as a series of phases and NVMe/NVMe-MI Contexts. The phases of the NVMe-MI Send command servicing model are further described in this section. The behavior of the portions of the figure in the NVMe Context are specified by the NVM Express Base Specification. The behavior of the portions of the figure in the NVMe-MI Context are specified by this specification. The phases and NVMe/NVMe-MI Contexts are logical constructs that illustrate the NVMe-MI Send command servicing model and do not mandate a particular implementation.

This section describes the NVMe-MI Send command servicing model starting at NVMe Processing as shown in phase 1 of Figure 50. In phase 1, CDW0 to CDW9 are checked for errors per the NVM Express Base Specification. If any errors are encountered in CDW0 to CDW9, then the NVMe-MI Send command is completed with an error status code in the Status Field as per the NVM Express Base Specification and the Tunneled Status and Tunneled NVMe Management Response fields shall be cleared to 0h.

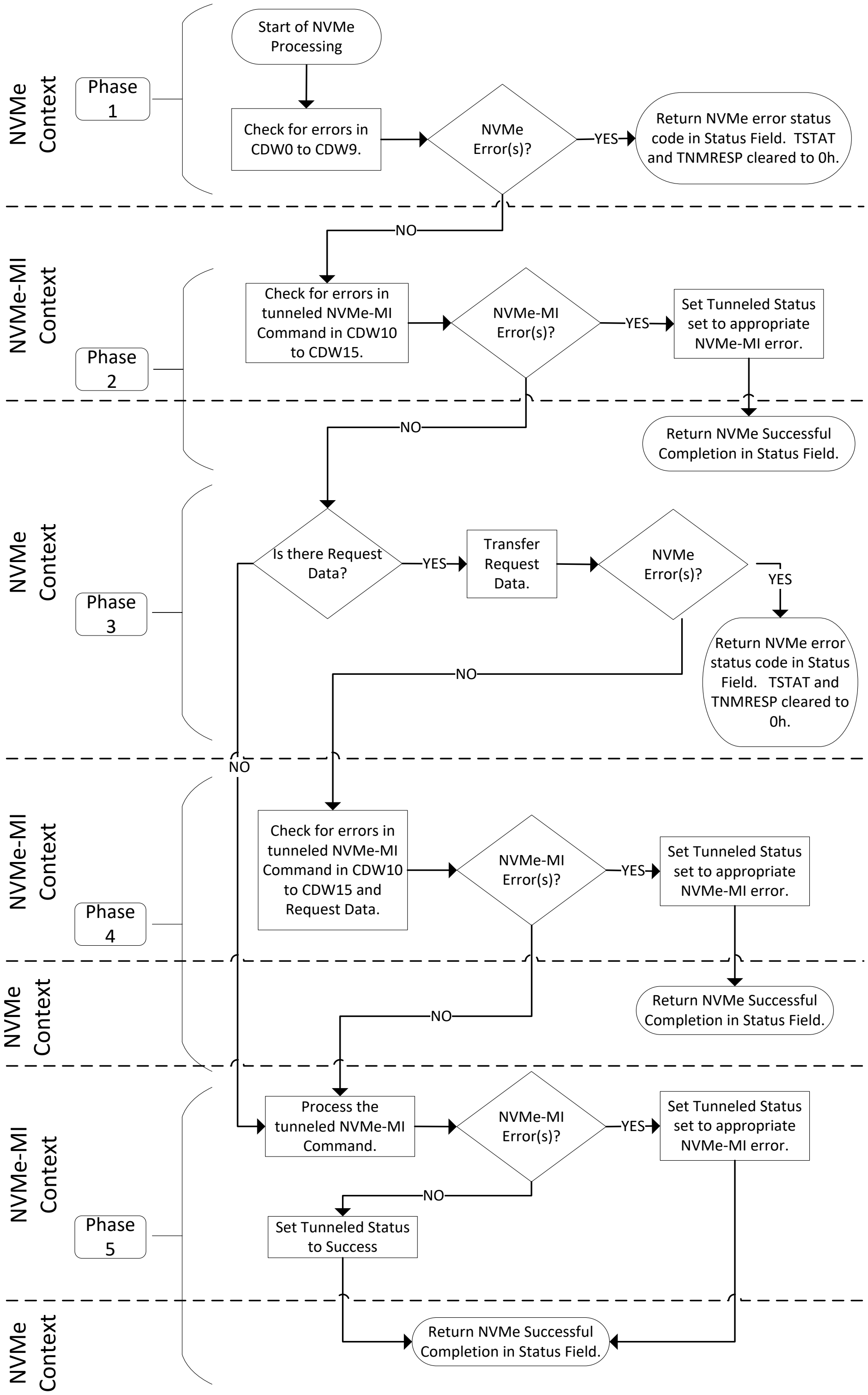
If there are no errors in CDW0 to CDW9, then command servicing enters phase 2 where the portion of the tunneled NVMe-MI Command in CDW10 to CDW15 is checked for errors. Note that if there is no Request Data, then CDW10 to CDW15 contain the entire tunneled NVMe-MI Command. If any errors are encountered in the portion of the tunneled NVMe-MI Command in CDW10 to CDW15, then the NVMe-MI Send command is completed with a status code of Successful Completion in the Status Field as defined in the NVM Express Base Specification. The Tunneled Status field contains the error Response Message Status for the portion of the tunneled NVMe-MI Command in CDW10 to CDW15 and the Tunneled NVMe Management Response field contains the NVMe Management Response field from the NVMe-MI Command that is being tunneled in-band.

If there are no errors in phase 2, then command servicing enters phase 3 where there is a check to determine if there is any Request Data for the tunneled NVMe-MI Command. If there is no Request Data for the tunneled NVMe-MI Command, then command servicing skips to phase 5. If there is Request Data, then the Request Data is transferred from the buffer pointed to by DPTR. If any errors are encountered transferring the Request Data, then the command is completed with an error status code in the Status Field as per the NVM Express Base Specification and the Tunneled Status and Tunneled NVMe Management Response fields shall be cleared to 0h.

If there are no errors transferring the data, then command servicing enters phase 4 where the whole tunneled NVMe-MI Command is constructed from CDW10 to CDW15 and the Request Data that was transferred. If any errors are encountered in the tunneled NVMe-MI Command, then the NVMe-MI Send command is completed with a status code of Successful Completion in the Status Field as defined in the NVM Express Base Specification. The Tunneled Status field contains the appropriate error Response Message Status and the Tunneled NVMe Management Response field contains the NVMe Management Response field from the NVMe-MI Command that is being tunneled in-band.

If there are no errors in phase 4, then command servicing enters phase 5 where the tunneled NVMe-MI Command finishes processing. If any errors are encountered processing the tunneled NVMe-MI Command, then the NVMe-MI Send command is completed with a status code of Successful Completion in the Status Field as defined in the NVM Express Base Specification and the Tunneled Status field contains the appropriate error Response Message Status. If the tunneled NVMe-MI Command is processed successfully, then the NVMe-MI Send command is completed with a status code of Successful Completion in the Status Field as defined in the NVM Express Base Specification. The Tunneled Status field contains a Response Message Status of Success for the tunneled NVMe-MI Command and the Tunneled NVMe Management Response field contains the NVMe Management Response field from the NVMe-MI Command that is being tunneled in-band.

Figure 50: NVMe-MI Send Command Servicing Model


NVMe Context
Phase 1
Start of NVMe Processing
Check for errors in CDW0 to CDW9.
NVMe Error(s)?
YES
Return NVMe error status code in Status Field. TSTAT and TNMRESP cleared to 0h.
NO
NVMe-MI Context
Phase 2
Check for errors in tunneled NVMe-MI Command in CDW10 to CDW15.
NVMe-MI Error(s)?
YES
Set Tunneled Status set to appropriate NVMe-MI error.
Return NVMe Successful Completion in Status Field.
NO
NVMe Context
Phase 3
Is there Request Data?
YES
Transfer Request Data.
NVMe Error(s)?
YES
Return NVMe error status code in Status Field. TSTAT and TNMRESP cleared to 0h.
NO
NO
NVMe-MI Context
Phase 4
Check for errors in tunneled NVMe-MI Command in CDW10 to CDW15 and Request Data.
NVMe-MI Error(s)?
YES
Set Tunneled Status set to appropriate NVMe-MI error.
NVMe Context
Return NVMe Successful Completion in Status Field.
NO
NVMe-MI Context
Phase 5
Process the tunneled NVMe-MI Command.
NVMe-MI Error(s)?
YES
Set Tunneled Status set to appropriate NVMe-MI error.
NO
Set Tunneled Status to Success
NVMe Context
Return NVMe Successful Completion in Status Field.

4.3.2 NVMe-MI Receive Command

The NVMe-MI Receive command is an NVMe Admin Command as defined by this specification and the NVM Express Base Specification. It is used to tunnel an NVMe-MI Command in-band from host software to an NVMe Controller that transfers data from an NVMe Controller to a host (similar to a read operation). The data being transferred is in one or more of the following locations: Response Data, NVMe Management Response. Figure 60 specifies which NVMe-MI Commands are tunneled via the NVMe-MI Receive command.

NVMe-MI Commands may apply to the NVM Subsystem, Controllers, and/or Namespaces. If the tunneled NVMe-MI Command applies to one or more Controllers, then the applicable Controller(s) are specified by fields in the tunneled NVMe-MI Command. Note that unlike some other Admin Commands, the Controller to which the NVMe-MI Receive command is issued is not used to determine which Controller the tunneled NVMe-MI Command applies to. If the tunneled NVMe-MI Command applies to one or more Namespaces, then the applicable Namespace(s) are specified by fields in the tunneled NVMe-MI Command. Note that the Namespace Identifier (NSID) field of the NVMe-MI Receive command (bytes 7:4 of the Submission Queue Entry) is not used and should be cleared to 0h by host software.

The mapping of how an NVMe-MI Command is tunneled inside of an NVMe-MI Receive command is described in section 4.3.2.1. The NVMe-MI Receive command servicing model is described in section 4.3.2.2.

4.3.2.1 NVMe-MI Receive Command Request Message to NVMe Admin Command SQE Mapping

In order to tunnel an NVMe-MI Command in-band via NVMe-MI Receive, an NVMe-MI Request Message is mapped onto an NVMe Submission Queue Entry (SQE) as shown pictorially in Figure 51 and in table form in Figure 52. An NVMe-MI Response Message is mapped on to an NVMe Completion Queue Entry (CQE) as shown pictorially in Figure 51 and in table form in Figure 53. Refer to the NVM Express Base Specification for details on an NVMe Submission Queue Entry and NVMe Completion Queue Entry.

Figure 51: NVMe-MI Receive Command Request/Response Message to NVMe Admin Command SQE/CQE Mapping Diagram

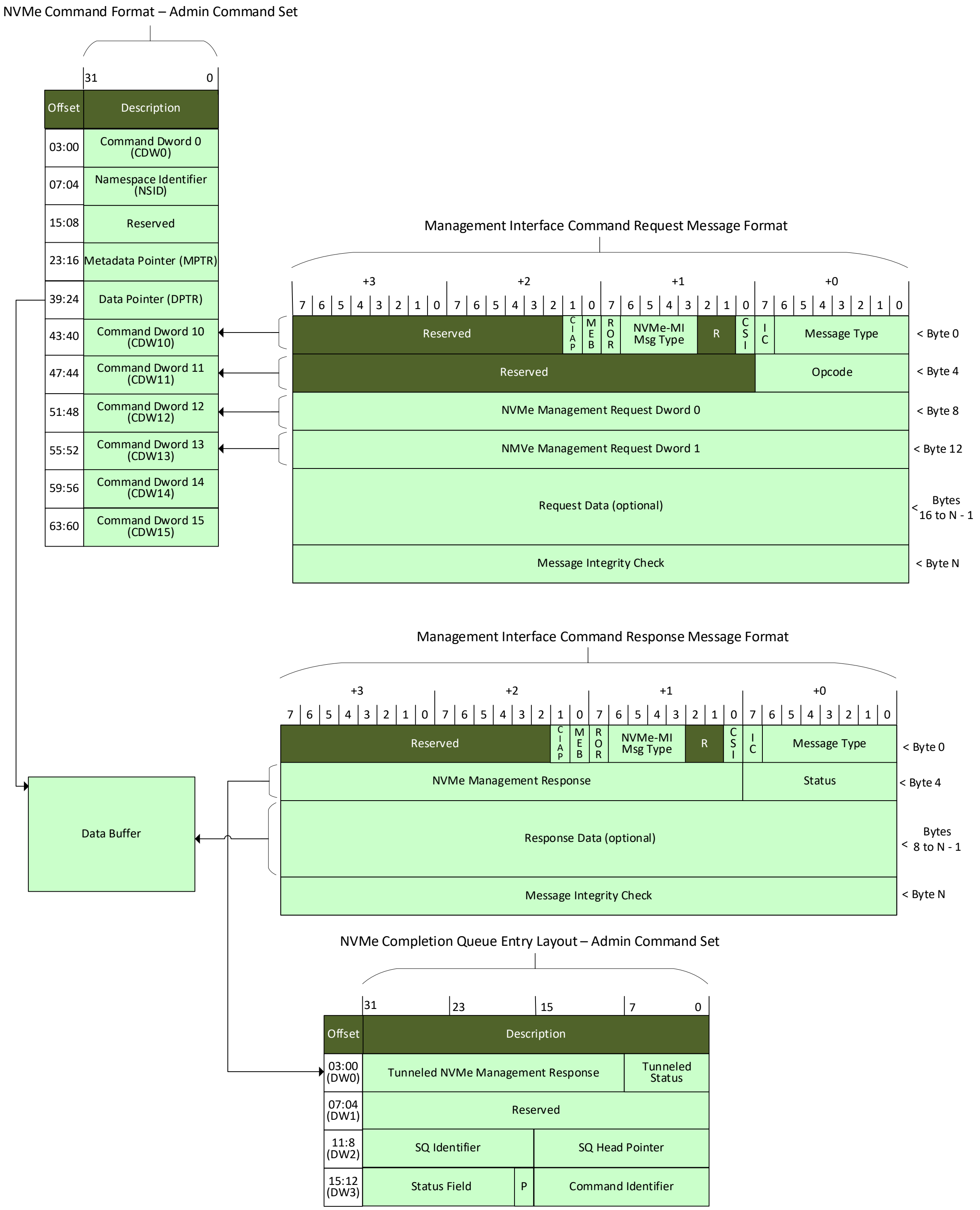


Figure 52: NVMe-MI Receive Command Request/Response Message to NVMe Admin Command SQE/CQE Mapping Table

NVMe-MI Command Request Message		NVMe Admin Command SQE Mapping	
Bytes	**Description**	**Bytes**	**Description**
n/a	This field has no equivalent in this specification.	03:00	**Command Dword 0 (CDW0):** Refer to the NVM Express Base Specification.
n/a	If the tunneled NVMe-MI Command requires one or more Namespaces to be specified, then the applicable Namespace Identifiers are specified by the tunneled NVMe-MI Command.	07:04	**Namespace Identifier (NSID):** This field should be cleared to 0h by host software. Refer to the NVM Express Base Specification for more details.
n/a	These bytes have no equivalent in this specification.	23:08	Refer to the NVM Express Base Specification.
n/a	There is no equivalent of DPTR in this specification. In NVMe-MI Receive, the Response Data is included in the Response Data portion of the Response Message.	39:24	**Data Pointer (DPTR):** This field contains a pointer to the start of the data buffer that contains the Response Data portion of the NVMe-MI Command that is being tunneled. If there is no Response Data for this command, then this field is ignored. Refer to the NVM Express Base Specification for the definition of this field.
03:00	NVMe-MI Message Header (NMH)	43:40	**Command Dword 10 (CDW10):** Dword 0 of the Request Message (NMH) that is being tunneled maps to CDW10 of the SQE. The byte ordering within CDW10 is little endian (i.e., NMH byte 0 maps to CDW10 byte 0, NMH byte 1 maps to CDW10 byte 1, etc.).
04	Opcode (OPC)	47:44	**Command Dword 11 (CDW11):** Dword 1 of the Request Message (OPC and Reserved bytes 7:5) that is being tunneled maps to CDW11 of the SQE. The byte ordering within CDW11 is little endian (i.e., OPC maps to CDW11 byte 0, the LSB of the Reserved field (NVMe-MI Command Request Message byte 5) maps to CDW11 byte 1, etc.).
07:05	Reserved		
11:08	NVMe Management Dword 0 (NMD0)	51:48	**Command Dword 12 (CDW12):** Dword 2 of the Request Message (NMD0) that is being tunneled maps to CDW12 of the SQE. The byte ordering within CDW12 is little endian (i.e., NMD0 byte 0 maps to CDW12 byte 0, NMD0 byte 1 maps to CDW12 byte 1).
15:12	NVMe Management Dword 1 (NMD1)	55:52	**Command Dword 13 (CDW13):** Dword 3 of the Request Message (NMD1) that is being tunneled maps to CDW13 of the SQE. The byte ordering within CDW13 is little endian (i.e., NMD1 byte 0 maps to CDW13 byte 0, NMD1 byte 1 maps to CDW13 byte 1).
n/a	This field has no equivalent in this specification.	59:56	**Command Dword 14 (CDW14):** Reserved.
n/a	This field has no equivalent in this specification.	63:60	**Command Dword 15 (CDW15):** Reserved.
N-1:16	Request Data (REQD)	n/a	There is no Request Data for NVMe-MI Receive.
N+3:N	Message Integrity Check (MIC)	n/a	The Message Integrity Check is not used in the in-band tunneling mechanism.

Figure 53: NVMe-MI Receive Command Response Message to NVMe Admin Command CQE Mapping Table

NVMe-MI Command Response Message		NVMe Admin Command CQE	
Bytes	**Description**	**Bytes**	**Description**
00	MCTP Data (MCTPD)	n/a	This field has no equivalent in the NVMe Admin Command CQE.
01	NVMe-MI Message Parameters (NMP)	n/a	This field has no equivalent in the NVMe Admin Command CQE.
03:02	Reserved	n/a	This field has no equivalent in the NVMe Admin Command CQE.
04	Status (STATUS)	03:00	**Command Specific (DW0):** Dword 1 of the Response Message (STATUS and NMRESP) that is being tunneled maps to DW0 of the CQE. The byte ordering within DW0 is little endian (i.e., STATUS maps to DW0 byte 0, the LSB of the NMRESP field (NVMe-MI Command Response Message byte 5) maps to DW0 byte 1, etc.). Refer to Figure 54 for additional details on this field.
07:05	NVMe Management Response (NMRESP)		
N-1:8	Response Data (RESPD)	n/a	Response Data is placed by the NVMe Controller into the data buffer pointed to by DPTR. If the Response Data size is not dword granular, then the Response Data shall be padded with the minimum number of bytes cleared to 0h to make the Response Data dword granular. The byte ordering within the data buffer pointed to by DPTR is little endian (i.e., RESPD byte 0 maps to byte 0 of the data buffer pointed to by DPTR, RESPD byte 1 maps to byte 1 of the data buffer pointed to by DPTR, etc.).
N+3:N	Message Integrity Check (MIC)	n/a	The Message Integrity Check is not used in the in-band tunneling mechanism.
n/a	These bytes have no equivalent in this specification.	15:04	Refer to the NVM Express Base Specification.

The definition of Dword 0 of the completion queue entry is in Figure 54.

Figure 54: NVMe-MI Receive – Completion Queue Entry Dword 0 (NRCQED0)

Bytes	Description
31:08	**Tunneled NVMe Management Response (TNMRESP):** This field contains the NVMe Management Response field from the NVMe-MI Command that is being tunneled in-band. If any errors are detected in the NVMe Context as described in section 4.3.2.2, then this field shall be cleared to 0h.
07:00	**Tunneled Status (TSTAT):** This field contains the Status field from the NVMe-MI Command that is being tunneled in-band. If any errors are detected in the NVMe Context as described in section 4.3.2.2, then this field shall be cleared to 0h.

4.3.2.2 NVMe-MI Receive Command Servicing Model

The NVMe-MI Receive command servicing model is illustrated in Figure 55 as a series of phases (described in this section) and NVMe/NVMe-MI Contexts. The phases of the NVMe-MI Receive command servicing model are further described in this section. The behavior of the portions of the figure in the NVMe Context are specified by the NVM Express Base Specification. The behavior of the portions of the figure in the NVMe-MI Context are specified by this specification. The phases and NVMe/NVMe-MI Contexts are logical constructs that illustrate the NVMe-MI Receive command servicing model and do not mandate a particular implementation.

This section describes the NVMe-MI Receive command servicing model starting at NVMe Processing as shown in phase 1 of Figure 55. In phase 1, CDW0 to CDW9 are checked for errors per the NVM Express Base Specification. If any errors are encountered in CDW0 to CDW9, then the command is completed with an error status code in the Status Field as per the NVM Express Base Specification and the Tunneled Status and Tunneled NVMe Management Response fields shall be cleared to 0h.

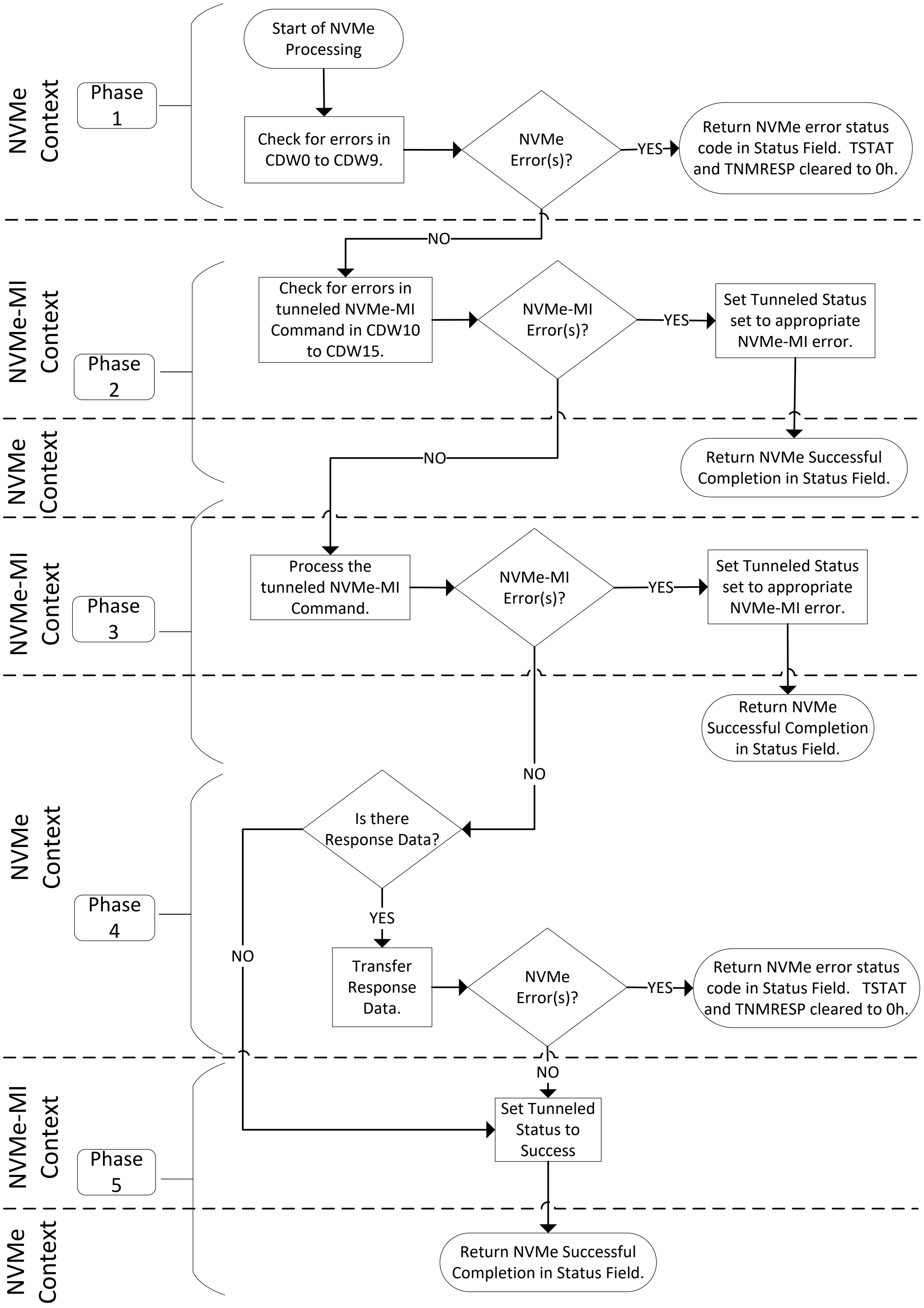
If there are no errors in CDW0 to CDW9, then command servicing enters phase 2 where the tunneled NVMe-MI Command in CDW10 to CDW15 is checked for errors. If any errors are encountered in the tunneled NVMe-MI Command in CDW10 to CDW15, then the NVMe-MI Receive command is completed with a status code of Successful Completion in the Status Field as defined in the NVM Express Base Specification. The Tunneled Status field contains the appropriate error Response Message Status and the Tunneled NVMe Management Response field contains the NVMe Management Response field from the NVMe-MI Command that is being tunneled in-band.

If there are no errors in phase 2, then command servicing enters phase 3 where the tunneled NVMe-MI Command finishes processing. If any errors are encountered processing the tunneled NVMe-MI Command, then the NVMe-MI Receive command is completed with a status code of Successful Completion in the Status Field as defined in the NVM Express Base Specification. The Tunneled Status field contains the appropriate error Response Message Status and the Tunneled NVMe Management Response field contains the NVMe Management Response field from the NVMe-MI Command that is being tunneled in-band.

If there are no errors in phase 3, then command servicing enters phase 4 where there is a check to determine if there is any Response Data for the tunneled NVMe-MI Command. If there is no Response Data for the tunneled NVMe-MI Command, then command servicing skips to phase 5. If there is Response Data, then the Response Data is transferred to the buffer pointed to by DPTR. If any errors are encountered transferring the Response Data then the command is completed with an error status code in the Status Field as per the NVM Express Base Specification and the Tunneled Status and Tunneled NVMe Management Response fields shall be cleared to 0h.

If there are no errors in phase 4, then command servicing enters phase 5 where the NVMe-MI Receive command is completed with a status code of Successful Completion in the Status Field as defined in the NVM Express Base Specification. The Tunneled Status field contains a Response Message Status of Success for the tunneled NVMe-MI Command and the Tunneled NVMe Management Response field contains the NVMe Management Response field from the NVMe-MI Command that is being tunneled in-band.

Figure 55: NVMe-MI Receive Command Servicing Model


NVMe Context
Phase 1
Start of NVMe Processing
Check for errors in CDW0 to CDW9.
NVMe Error(s)?
YES
Return NVMe error status code in Status Field. TSTAT and TNMRESP cleared to 0h.
NO
NVMe-MI Context
Phase 2
Check for errors in tunneled NVMe-MI Command in CDW10 to CDW15.
NVMe-MI Error(s)?
YES
Set Tunneled Status set to appropriate NVMe-MI error.
NVMe Context
Return NVMe Successful Completion in Status Field.
NO
NVMe-MI Context
Phase 3
Process the tunneled NVMe-MI Command.
NVMe-MI Error(s)?
YES
Set Tunneled Status set to appropriate NVMe-MI error.
Return NVMe Successful Completion in Status Field.
NO
NVMe Context
Phase 4
Is there Response Data?
YES
Transfer Response Data.
NVMe Error(s)?
YES
Return NVMe error status code in Status Field. TSTAT and TNMRESP cleared to 0h.
NO
NO
NVMe-MI Context
Phase 5
Set Tunneled Status to Success
NVMe Context
Return NVMe Successful Completion in Status Field.

5 Management Interface Command Set

The Management Interface Command Set defines the Command Messages that may be submitted by a Requester when the NMIMT value is set to NVMe-MI Command. The Management Interface Command Set is applicable to both the out-of-band mechanism and the in-band tunneling mechanism. The processing of commands in the Management Interface Command Set may be affected by the Command and Feature Lockdown feature (refer to the NVM Express Base Specification).

The NVMe-MI Message structure with all fields that are common to all NVMe-MI Messages is defined in section 3.1. The Response Message structure for the Management Interface Command Set is defined in section 4.1.2. The Message Body for the Management Interface Command Set is shown in Figure 56 and Figure 57. Command specific fields for the Management Interface Command Set are defined in this section.

Figure 56: NVMe-MI Command Request Message Format

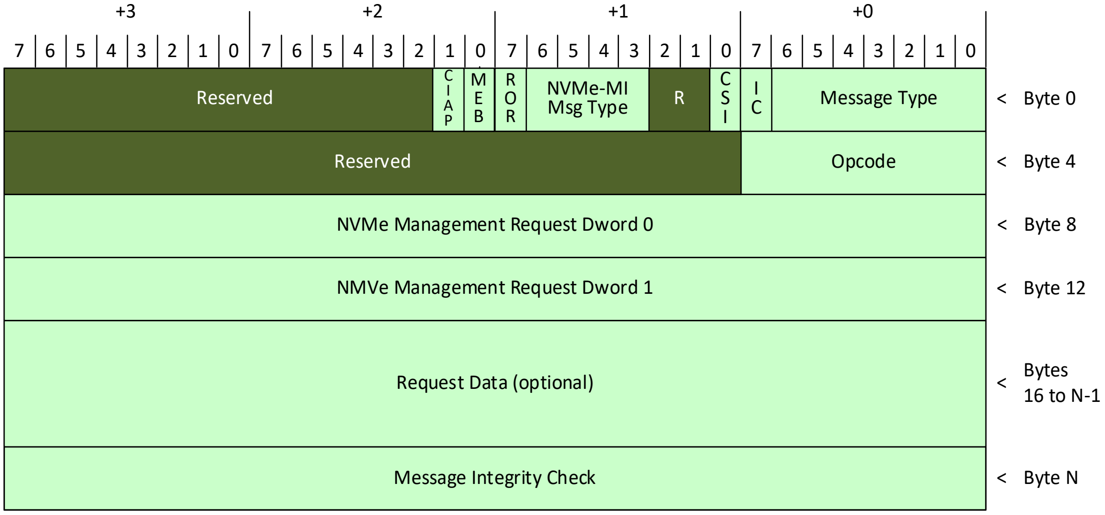


Figure 57: NVMe-MI Command Request Message Description (NCREQ)

Bytes	Description
03:00	**NVMe-MI Message Header (NMH):** Refer to section 3.1.
04	**Opcode (OPC):** This field specifies the opcode of the NVMe-MI Command to be processed. Refer to Figure 58.
07:05	Reserved
11:08	**NVMe Management Dword 0 (NMD0):** This field is command specific Dword 0.
15:12	**NVMe Management Dword 1 (NMD1):** This field is command specific Dword 1.
N-1:16	**Request Data (REQD):** (Optional)
N+3:N	**Message Integrity Check (MIC):** Refer to section 3.1.

The Request Data field is an optional field included in some NVMe-MI Commands. If the size of the Request Data does not match the specified Data Length of the Command Message, then the Responder responds with a Generic Error Response and Invalid Command Input Data Size status.

Figure 58 defines the Management Interface Command Set opcodes.

Figure 58: Opcodes for Management Interface Command Set

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
0Ch	Shutdown
0Dh to BFh	Reserved
C0h to FFh	Vendor specific

Figure 59 shows the Management Interface Command Set commands that are mandatory, optional, and prohibited for an NVMe Storage Device and for an NVMe Enclosure using the out-of-band mechanism. Figure 60 shows Management Interface Command Set commands that are mandatory, optional, and prohibited for an NVMe Storage Device and for an NVMe Enclosure using the in-band tunneling mechanism.

Figure 59: Management Interface Command Set 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
O	O[3]	VPD Write
O	O[3]	Reset
P	M	SES Receive
P	M	SES Send
O	O[3]	Shutdown
O	M	Management Endpoint Buffer Read
O	M	Management Endpoint Buffer Write

Figure 59: Management Interface Command Set Support using an Out-of-Band Mechanism

NVMe Storage Device O/M/P[1]	NVMe Enclosure O/M/P[1]	Command
O	O	Vendor specific

NOTES:

1. O/M/P 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 or an NVMe Enclosure and may implement optional commands allowed by either an NVMe Storage Device or 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 60: Management Interface Command Set Support using In-Band Tunneling Mechanism

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

NOTES:

1. O/M/P 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 or an NVMe Enclosure and may implement optional commands allowed by either an NVMe Storage Device or 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.
3. This column indicates whether the NVMe-MI Command is tunneled in-band using the NVMe-MI Send or NVMe-MI Receive command.

Figure 61: NVMe-MI Command Response Message Format

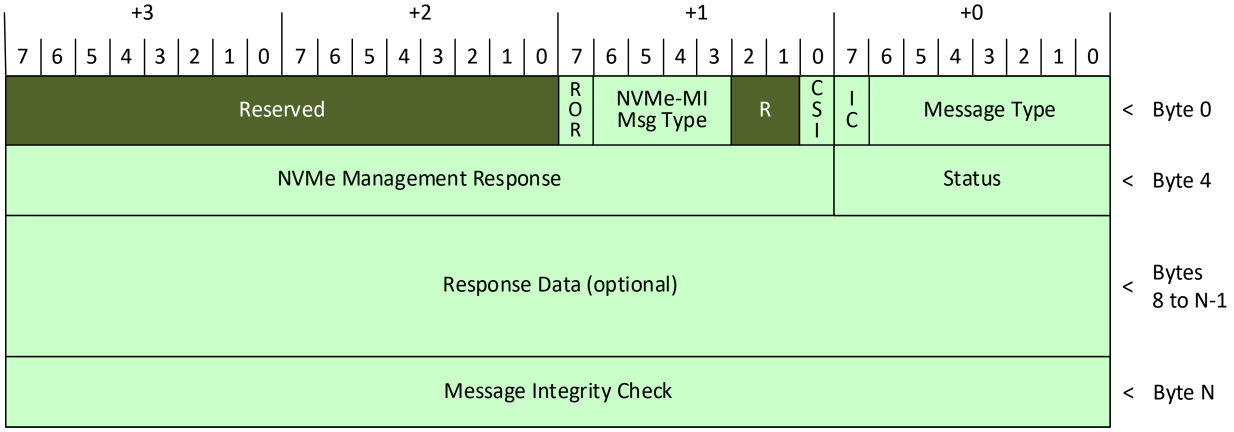


Figure 62: NVMe-MI Command Response Message Description (NCRESP)

Bytes	Description
03:00	**NVMe-MI Message Header (NMH):** Refer to section 3.1.
04	**Status (STATUS):** This field indicates the status of the NVMe-MI Command. Refer to section 4.1.2.
07:05	**NVMe Management Response (NMRESP):** This field is command specific.
N-1:08	**Response Data (RESPD):** (Optional)
N+3:N	**Message Integrity Check (MIC):** Refer to section 3.1.

5.1 Configuration Get

The Configuration Get command allows the Requester to read the current configuration of a Responder.

The command uses NVMe Management Dwords 0 and 1. The format of NVMe Management Dwords 0 and 1 are shown in Figure 63 and Figure 64 respectively. There is no Request Data included in a Configuration Get command.

Figure 63: Configuration Get – NVMe Management Dword 0

Bits	Description
31:08	Configuration Identifier specific
07:00	**Configuration Identifier:** This field specifies the identifier of the Configuration that is being read. Refer to Figure 65.

Figure 64: Configuration Get – NVMe Management Dword 1

Bits	Description
31:00	Configuration Identifier specific

NVMe-MI Configuration Identifiers are listed in Figure 65.

Figure 65: NVMe Management Interface Configuration Identifiers

Configuration Identifier	Out-of-Band Mechanism O/M/P[1]	In-Band Tunneling Mechanism O/M/P[1]	Description
00h	-	-	Reserved
01h	M	P	SMBus/I2C Frequency
02h	M	M	Health Status Change
03h	M	P	MCTP Transmission Unit Size
04h to BFh	-	-	Reserved
C0h to FFh	O	O	Vendor Specific
NOTES: 1. O/M/P definition: O = Optional, M = Mandatory, P = Prohibited from being supported.			

The NVMe Management Response field is configuration specific.

5.1.1 SMBus/I2C Frequency (Configuration Identifier 01h)

The SMBus/I2C Frequency configuration indicates the current frequency of the SMBus port, if applicable.

The configuration specific fields in NVMe Management Dword 0 are shown in Figure 66. The configuration specific fields in NVMe Management Dword 1 are reserved. The current SMBus/I2C Frequency configuration is returned in the NVMe Management Response field as shown in Figure 67.

Figure 66: SMBus/I2C Frequency – NVMe Management Dword 0

Bits	Description
31:24	**Port Identifier:** This field specifies the port whose SMBus/I2C Frequency is indicated.
23:08	Reserved
07:00	**Configuration Identifier:** This field specifies the identifier of the Configuration that is being read. Refer to Figure 65.

Figure 67: SMBus/I2C Frequency – NVMe Management Response

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>23:04</td><td>Reserved</td></tr>
<tr><td>03:00</td><td><b>SMBus/I2C Frequency:</b> The current frequency of the SMBus/I2C. The default value for this field following a reset or power cycle is 1h, if SMBus is supported.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>SMBus is not supported or is disabled</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>4h to Fh</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>

5.1.2 Health Status Change (Configuration Identifier 02h)

The Health Status Change configuration is used to clear the selected status bits in the Composite Controller Status field using Configuration Set. A Requester should not use Configuration Get for this Configuration Identifier.

The configuration specific fields in NVMe Management Dwords 0 and 1 are reserved. A Responder shall complete a Configuration Get command on this Configuration Identifier with a Success Response. The NVMe Management Response field is reserved and there is no Response Data.

5.1.3 MCTP Transmission Unit Size (Configuration Identifier 03h)

The MCTP Transmission Unit Size configuration indicates the current MCTP Transmission Unit Size of the Port Identifier specified in Dword 0.

The configuration specific fields in NVMe Management Dword 0 are shown in Figure 68. The configuration specific fields in NVMe Management Dword 1 are reserved. The current Transmission unit size of the specified port is returned in the NVMe Management Response field as shown in Figure 69.

Figure 68: MCTP Transmission Unit Size – NVMe Management Dword 0

Bits	Description
31:24	**Port Identifier:** This field specifies the port whose MCTP Transmission Unit Size is indicated.
23:08	Reserved
07:00	**Configuration Identifier:** This field specifies the identifier of the Configuration that is being read. Refer to Figure 65.

Figure 69: MCTP Transmission Unit Size – NVMe Management Response

Bits	Description
23:16	Reserved
15:00	**MCTP Transmission Unit Size:** This field contains the MCTP Transmission Unit Size in bytes to be used by the port. The default value for this field following a reset or power cycle is 40h (64).

5.2 Configuration Set

The Configuration Set command allows the Requester to modify the current configuration of a Responder.

The command uses NVMe Management Dwords 0 and 1. The format of NVMe Management Dwords 0 and 1 are shown in Figure 70 and Figure 71 respectively. There is no Request Data included in a Configuration Set command.

Figure 70: Configuration Set – NVMe Management Dword 0

Bits	Description
31:08	Configuration Identifier specific
07:00	**Configuration Identifier:** This field specifies the identifier of the Configuration that is being written. Refer to Figure 65.

Figure 71: Configuration Set – NVMe Management Dword 1

Bits	Description
31:00	Configuration Identifier specific

NVMe-MI Configuration Identifiers are listed in Figure 65. Specifying a reserved identifier in the Configuration Identifier field causes the command to complete with an Invalid Parameter Error Response with the PEL field indicating the Configuration Identifier field.

The NVMe Management Response field is configuration Identifier specific.

5.2.1 SMBus/I2C Frequency (Configuration Identifier 01h)

The SMBus/I2C Frequency configuration specifies a new frequency for the SMBus port.

The configuration specific fields in NVMe Management Dword 0 are shown in Figure 72. The configuration specific fields in NVMe Management Dword 1 are reserved. NVMe Management Response field is reserved.

After successful completion of this command, the SMBus/I2C frequency is updated to the specified frequency. A Management Controller should not change this configuration while there are other Command Messages outstanding.

If the specified frequency is not supported, then the Management Endpoint shall respond with an Invalid Parameter Error Response with the PEL field indicating the SMBus/I2C Frequency field. If the Port Identifier specified is not an SMBus/I2C port, then the Management Endpoint shall respond with an Invalid Parameter Error Response with the PEL field indicating the Port Identifier field.

Figure 72: SMBus/I2C Frequency – NVMe Management Dword 0

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:24</td><td><b>Port Identifier:</b> This field specifies the port whose SMBus/I2C Frequency is specified.</td></tr>
<tr><td>23:12</td><td>Reserved</td></tr>
<tr><td>11:08</td><td><b>SMBus/I2C Frequency:</b> This field specifies the new frequency for the specified SMBus/I2C port.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>Reserved</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>4h to Fh</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>07:00</td><td><b>Configuration Identifier:</b> This field specifies the identifier of the Configuration that is being written. Refer to Figure 65.</td></tr>
</table>

5.2.2 Health Status Change (Configuration Identifier 02h)

This Configuration Identifier is used to clear selected status bits in the Composite Controller Status field of the NVM Subsystem Health Data Structure (refer to Figure 90) returned by the NVM Subsystem Health Status Poll command.

The Composite Controller Status field of the NVM Subsystem Health Data Structure is used to report the occurrence of health and status events associated with the NVM Subsystem. When a bit in this field is set to ‘1’, it remains a ‘1’ until cleared by a Requester.

A Configuration Set command that selects Health Status Change may be used to clear corresponding bits selected in NVMe Management Dword 1 of the Composite Controller Status field to ‘0’.

A Configuration Set command that selects Health Status Change operates independently in the out-of-band mechanism and the in-band tunneling mechanism.

An NVMe Storage Device or NVMe Enclosure supporting the Health Status Change Configuration Identifier in the out-of-band mechanism shall have an independent copy of the Composite Controller Status dedicated to the out-of-band mechanism. In the out-of-band mechanism, a Configuration Set command that selects Health Status Change only applies to the copy of the Composite Controller Status dedicated to the out-of-band mechanism. Refer to section 5.4 for more details on Composite Controller Status.

An NVMe Storage Device or NVMe Enclosure supporting the Health Status Change Configuration Identifier in the in-band tunneling mechanism shall have an independent copy of the Composite Controller Status dedicated to the in-band tunneling mechanism. In the in-band tunneling mechanism, a Configuration Set command that selects Health Status Change only applies to the copy of the Composite Controller Status dedicated to the in-band tunneling mechanism.

Figure 73: Health Status Change - NVMe Management Dword 0

Bits	Description
31:08	Reserved
07:00	**Configuration Identifier:** This field specifies the identifier of the Configuration that is being written. Refer to Figure 65.

Figure 74: Health Status Change – NVMe Management Dword 1

Bits	Description
31:12	Reserved
11	**Critical Warning (CWARN):** When this bit is set to ‘1’, bit 12 in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to ‘0’.
10	**Available Spare (SPARE):** When this bit is set to ‘1’, bit 11 in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to ‘0’.
09	**Percentage Used (PDLU):** When this bit is set to ‘1’, bit 10 in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to ‘0’.
08	**Composite Temperature (CTEMP):** When this bit is set to ‘1’, bit 9 in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to ‘0’.
07	**Controller Status Change (CSCHNG):** When this bit is set to ‘1’, bit 8 in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to ‘0’.
06	**Firmware Activated (FA):** When this bit is set to ‘1’, bit 7 in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to ‘0’.
05	**Namespace Attribute Changed (NAC):** When this bit is set to ‘1’, bit 6 in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to ‘0’.
04	**Controller Enable Change Occurred (CECO):** When this bit is set to ‘1’, bit 5 in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to ‘0’.
03	**NVM Subsystem Reset Occurred (NSSRO):** When this bit is set to ‘1’, bit 4 in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to ‘0’.
02	**Shutdown Status (SHST):** When this bit is set to ‘1’, bit 2 in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to ‘0’.
01	**Controller Fatal Status (CFS):** When this bit is set to ‘1’, bit 1 in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to ‘0’.
00	**Ready (RDY):** When this bit is set to ‘1’, bit 0 in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to ‘0’.

5.2.3 MCTP Transmission Unit Size (Configuration Identifier 03h)

The MCTP Transmission Unit Size configuration specifies a new MCTP Transmission Unit Size for the specified Port Identifier. A Management Controller should check the maximum MCTP Transmission Unit Size for the port reported by the Management Endpoint using the Read NVMe-MI Data Structure command (refer to Figure 95).

The configuration specific fields in NVMe Management Dwords 0 and 1 are shown in Figure 75 and Figure 76 respectively. The NVMe Management Response field is reserved.

After successful completion of this command, the MCTP Transmission Unit Size for MCTP packets on the specified port is updated to the specified size for future Command Messages. A Management Controller should not change this configuration while there are other commands outstanding. Changing this configuration while there are other Request Messages outstanding results in undefined behavior. If a Request Message is sent with a given MCTP Transmission Unit Size, then issuing a Replay Control Primitive after changing the MCTP Transmission Unit Size to a different value results in undefined behavior.

If the specified MCTP Transmission Unit Size is not supported, then the Management Endpoint shall abort the command and send a Response Message with an Invalid Parameter Error Response with the PEL field indicating the MCTP Transmission Unit Size field. If the Port Identifier specified is not valid, then the Management Endpoint shall abort the command and send a Response Message with an Invalid Parameter Error Response with the PEL field indicating the Port Identifier field.

Figure 75: MCTP Transmission Unit Size – NVMe Management Dword 0

Bits	Description
31:24	**Port Identifier:** This field specifies the port whose MCTP Transmission Unit Size is specified.
23:08	Reserved
07:00	**Configuration Identifier:** This field specifies the identifier of the Configuration that is being written. Refer to Figure 65.

Figure 76: MCTP Transmission Unit Size – NVMe Management Dword 1

Bits	Description
31:16	Reserved
15:00	**MCTP Transmission Unit Size:** This field contains the MCTP Transmission Unit Size in bytes to be used by the port.

5.3 Controller Health Status Poll

The Controller Health Status Poll command is used to efficiently determine changes in health status attributes associated with one or more Controllers in the NVM Subsystem.

The Controller Health Status Poll command operates independently in the out-of-band mechanism and the in-band tunneling mechanism.

An NVMe Storage Device or NVMe Enclosure supporting the Controller Health Status Poll command in the out-of-band mechanism shall have an independent copy of the Controller Health Data Structure (refer to Figure 80) and the Controller Health Status Changed Flags (refer to Figure 81) dedicated to the out-of-band mechanism. In the out-of-band mechanism, a Controller Health Status Poll command only applies to the copy of the Controller Health Data Structure and the Controller Health Status Changed Flags dedicated to the out-of-band mechanism.

An NVMe Storage Device or NVMe Enclosure supporting the Controller Health Status Poll command in the in-band tunneling mechanism shall have an independent copy of the Controller Health Data Structure and the Controller Health Status Changed Flags dedicated to the in-band tunneling mechanism. In the in-band tunneling mechanism, a Controller Health Status Poll command only applies to the copy of the Controller Health Data Structure and the Controller Health Status Changed Flags dedicated to the in-band tunneling mechanism.

The Controller Health Status Poll command uses NVMe Management Dwords 0 and 1. The format of NVMe Management Dword 0 is shown in Figure 77 and the format of NVMe Management Dword 1 is shown in Figure 78.

Figure 77: Controller Health Status Poll – NVMe Management Dword 0

Bits	Description
31	**Report All (ALL):** When this bit is set to ‘1’, a Controller Health Data Structure is returned regardless of the status of the Controller Health Status Changed Flags. The Controller selection fields (SCTLID, MAXRENT, INCF, INCPF, and INCVF) still apply even when this bit is set to ‘1’ but the error selection bits (CWARN, SPARE, PDLU, CTEMP, and CSTS in Figure 78) do not apply. When this bit is cleared to ‘0’, a Controller Health Data Structure is returned based on the Controller selection fields (SCTLID, MAXRENT, INCF, INCPF, and INCVF) and error selection fields (CWARN, SPARE, PDLU, CTEMP, and CSTS in Figure 78).
30:27	Reserved
26	**Include SR-IOV Virtual Functions (INCVF):** When this bit is set to ‘1’, a Controller Health Data Structure is returned for Controllers associated with SR-IOV Virtual Functions (VFs).
25	**Include SR-IOV Physical Functions (INCPF):** When this bit is set to ‘1’, a Controller Health Data Structure is returned for Controllers associated with SR-IOV Physical Functions (PFs).
24	**Include PCI Functions (INCF):** When this bit is set to ‘1’, a Controller Health Status Data Structure is returned for Controllers associated with non-SR-IOV PCI Functions.
23:16	**Maximum Response Entries (MAXRENT):** This field specifies the maximum number of Controller Health Data Structure entries that may be returned in the completion. This is a 0’s based field. The maximum number of entries is 255. Specifying 256 entries is interpreted as an Invalid Parameter.
15:00	**Starting Controller ID (SCTLID):** This field specifies the Controller ID of the first Controller whose Controller Health Data Structure may be returned.

Figure 78: Controller Health Status Poll – NVMe Management Dword 1

Bits	Description
31	**Clear Changed Flags (CCF):** When this bit is set to ‘1’, the Controller Health Status Changed Flags are cleared in Controllers whose Controller Health Data Structure is contained in the Response Data.
30:05	Reserved
04	**Critical Warning (CWARN):** When this bit is set to ‘1’, a Controller Health Data Structure is returned for Controllers with the Critical Warning bit set to ‘1’ in their Controller Health Status Changed Flags.
03	**Available Spare (SPARE):** When this bit is set to ‘1’, a Controller Health Data Structure is returned for Controllers with the Available Spare bit set to ‘1’ in their Controller Health Status Changed Flags.
02	**Percentage Used (PDLU):** When this bit is set to ‘1’, a Controller Health Data Structure is returned for Controllers with the Percent Used bit set to ‘1’ in their Controller Health Status Changed Flags.
01	**Composite Temperature Changes (CTEMP):** When this bit is set to ‘1’, a Controller Health Data Structure is returned for Controllers with the Composite Temperature bit set to ‘1’ in their Controller Health Status Changed Flags.
00	**Controller Status Changes (CSTS):** When this bit is set to ‘1’, a Controller Health Data Structure is returned for Controllers with the Ready, Controller Fatal Status, Shutdown Status, NVM Subsystem Reset Occurred, Controller Enable Change Occurred, Namespace Attribute Changed, or Firmware Activated bit set to ‘1’ in their Controller Health Status Changed Flags.

The Controller Health Status Poll Response Messages use the NVMe Management Response field with the format shown in Figure 79.

The Response Data field size may vary based on the number of Controllers whose Controller Health Data Structure has changed and based on the number of Controllers whose Controller Health Data Structure is filtered out by Controller type (refer to section 5.3.1) or Controller Health Status Changed Flags (refer to section 5.3.2). The Response Entries field indicates the number of Controller Health Data Structures that are contained in the Response Data.

Figure 79: Controller Health Status Poll – NVMe Management Response

Bits	Description
23:16	**Response Entries (RENT):** This field specifies the number of Controller Health Data Structure Entries present in the Response Data for this Response Message.
15:00	Reserved

The Controller Health Data Structure, shown in Figure 80, contains the health status attributes that are tracked for each Controller. When the command is processed without error, health status is returned for up to 255 Controllers starting at or above the Starting Controller ID (SCTLID). Controllers are returned in ascending order of Controller Identifier starting at offset 0h of the Response Data.

Figure 80: Controller Health Data Structure (CHDS)

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>01:00</td><td><b>Controller Identifier (CTLID):</b> This field specifies the Controller Identifier with which the data contained in this data structure is associated.</td></tr>
<tr><td>03:02</td><td><b>Controller Status (CSTS):</b> This field reports the Controller status.
<table>
<tr><th>Bits</th><th>Reset</th><th>Description</th></tr>
<tr><td>15:08</td><td>0</td><td>Reserved</td></tr>
<tr><td>07</td><td>HwInit</td><td><b>Firmware Activated (FA):</b> This bit is set to ‘1’ when a new firmware image is activated. Firmware activation is described in the NVM Express Base Specification.
The reset value of this bit is set to ‘1’ if a reset caused a new firmware image to be activated.</td></tr>
<tr><td>06</td><td>0</td><td><b>Namespace Attribute Changed (NAC):</b> This bit is set to ‘1’ under the same conditions that causes the Namespace Attribute Changed asynchronous event to be sent if Namespace Attribute Notices are enabled as specified in the NVM Express Base Specification. This bit may be set to ‘1’ regardless of whether Namespace Attribute Notices are enabled or not.</td></tr>
<tr><td>05</td><td>0</td><td><b>Controller Enable Change Occurred (CECO):</b> This bit is set to ‘1’ when the Enable bit (refer to CC.EN in the NVM Express Base Specification) changes state.</td></tr>
<tr><td>04</td><td>HwInit</td><td><b>NVM Subsystem Reset Occurred (NSSRO):</b> This bit corresponds to the value of the NVM Subsystem Reset Occurred (refer to CSTS.NSSRO in the NVM Express Base Specification) bit.</td></tr>
<tr><td>03:02</td><td>00b</td><td><b>Shutdown Status (SHST):</b> This field corresponds to the value of the Shutdown Status (refer to CSTS.SHST in the NVM Express Base Specification) field.</td></tr>
<tr><td>01</td><td>HwInit</td><td><b>Controller Fatal Status (CFS):</b> This bit corresponds to the value of the Controller Fatal Status (refer to CSTS.CFS in the NVM Express Base Specification) bit.</td></tr>
<tr><td>00</td><td>0</td><td><b>Ready (RDY):</b> This bit corresponds to the value of the Ready (refer to CSTS.RDY in the NVM Express Base Specification) bit.</td></tr>
</table></td></tr>
<tr><td>05:04</td><td><b>Composite Temperature (CTEMP):</b> This field contains a value corresponding to a temperature in Kelvins that represents the current composite temperature of the Controller and Namespace(s) associated with that Controller. The value of this field corresponds to the value in the Controller’s SMART / Health Information Log.</td></tr>
<tr><td>06</td><td><b>Percentage Used (PDLU):</b> This field contains a vendor specific estimate of the percentage of NVM Subsystem life used based on the actual usage and the manufacturer’s prediction of NVM life. The value of this field corresponds to the value in the Controller’s SMART / Health Information Log.</td></tr>
<tr><td>07</td><td><b>Available Spare (SPARE):</b> This field contains a normalized percentage (0% to 100%) of the remaining spare capacity available. The value of this field corresponds to the value in the Controller’s SMART / Health Information Log.</td></tr>
</table>

Figure 80: Controller Health Data Structure (CHDS)

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>08</td><td><b>Critical Warning (CWARN):</b> This field indicates critical warnings for the state of the Controller. The value of this field corresponds to the value in the Controller’s SMART / Health Information Log.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:5</td><td>Reserved</td></tr>
<tr><td>4</td><td><b>Volatile Memory Backup Failed (VMBF):</b> This bit is set to ‘1’ when the volatile memory backup device has failed.</td></tr>
<tr><td>3</td><td><b>Read Only (RO):</b> This bit is set to ‘1’ when the media has been placed in read only mode.</td></tr>
<tr><td>2</td><td><b>Reliability Degraded (RD):</b> This bit is set to ‘1’ when NVM Subsystem reliability has been degraded due to significant media related errors or an internal error.</td></tr>
<tr><td>1</td><td><b>Temperature Above or Under Threshold (TAUT):</b> This bit is set to ‘1’ when a temperature is above an over temperature threshold or below an under-temperature threshold.</td></tr>
<tr><td>0</td><td><b>Spare Threshold (ST):</b> This bit is set to ‘1’ when the available spare has fallen below the available spare threshold.</td></tr>
</table>
</td></tr>
<tr><td>15:09</td><td>Reserved</td></tr>
</table>

Associated with each Controller in the NVM Subsystem is a set of Controller Health Status Changed Flags shown in Figure 81. The Controller Health Status Changed Flags are set when the corresponding field in the Controller Health Data Structure changes state as described in Figure 81. Figure 82 shows a graphical representation of which field(s)/bit(s) in the Controller Health Data Structure are associated with each bit in the Controller Health Status Changed Flags. When a bit in the Controller Health Status Changed Flags for any Controller transitions from ‘0’ to ‘1’, then the corresponding bit in the Composite Controller Status is also set to ‘1’. The Controller Health Status Changed Flags are cleared in Controllers whose Controller Health Data Structure is returned in the Success Response to a Controller Health Status Poll Command Message with the Clear Changed Flags bit set to ‘1’.

A Controller Health Status Poll response may return the Controller Health Data Structure for up to 255 Controllers in the Response Data field. An NVM Subsystem may contain up to 64 Ki Controllers, so a method is required to limit the size of the Response Message. The Starting Controller ID field in the Command Message specifies the Controller ID of the first Controller whose Controller Health Data Structure may be returned in the Response Data field. The Maximum Response Entries field specifies the maximum number of Controllers whose Controller Health Data Structure may be returned in the Response Data field.

The Response Data field contains the Controller Health Status Data Structure for up to the first M Controllers starting with Controller N, where M is equal to the Maximum Response Entries field and N is equal to the Starting Controller ID field. The Response Data field shall contain the Controller Health Status Data Structure for all Controllers that do not match the filtering criteria in Controller Health Status Poll - NVMe Management Dword 0 (refer to section 5.3.1) and that have one or more Controller Health Status Changed Flags that are: a) set and b) do not match the filtering criteria in Controller Health Status Poll - NVMe Management Dword 1 (refer to section 5.3.2). The Response Data field shall not contain the Controller Health Status Data Structure for any Controllers that meet the filtering criteria in sections 5.3.1 or 5.3.2.

5.3.1 Filtering by Controller Type

The Controller Health Data Structures that are returned by Controller Health Status Poll may be filtered (i.e., excluded from being included in the Response Data field regardless of the state of the Controller Health Status Changed Flags) by Controller type (i.e., non SR-IOV PCI Function, SR-IOV PF, and SR-IOV VF). Controller type filtering is controlled by the Include PCI Functions, Include SR-IOV PFs, and Include

SR-IOV VFs fields in NVMe Management Dword 0. When one of these bits is set to ‘1’, Controller Health Data Structures for Controllers corresponding to that type of PCI Function are included in the Response Data field; else, the Controller Health Data Structure for that Controller is excluded from the Response Data field.

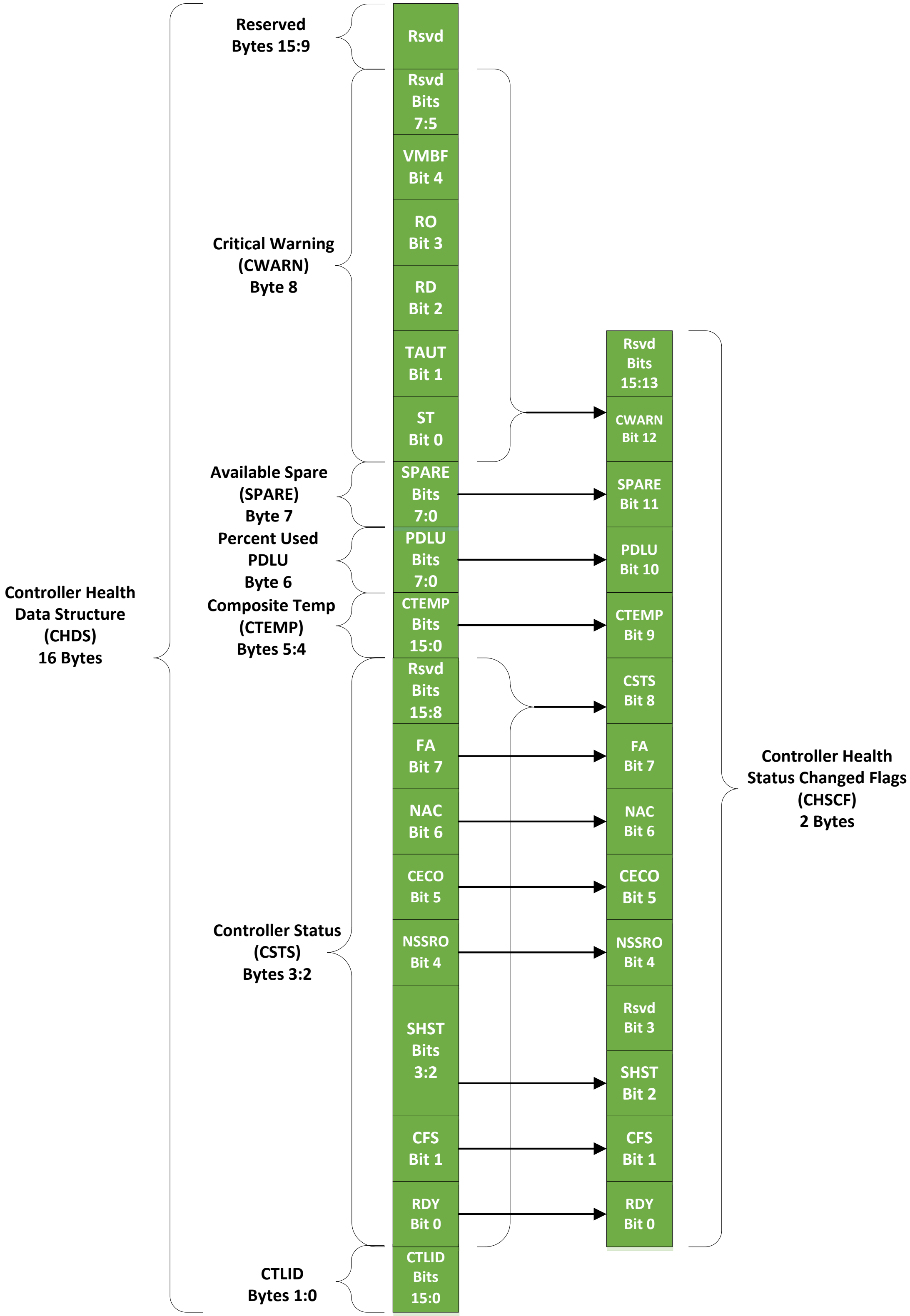
5.3.2 Filtering by Controller Health Status Changed Flags

The Controller Health Data Structures that are returned by Controller Health Status Poll may also be filtered by the Controller Health Status Changed Flags. Filtering of changes by Controller Health Status Changed Flags is controlled by some of the bits in NVMe Management Dword 1. When one or more of these bits are set to ‘1’ and any of the corresponding bit(s) in the Controller Health Status Changed Flags for the Controller are also set to ‘1’ (refer to Figure 78 for Controller Health Status Changed Flags associated with each bit in NVMe Management Dword 1), then the entire Controller Health Data Structure (including any filtered fields) for that Controller is returned in the Response Data field; else, the Controller Health Data Structure for that Controller is excluded from the Response Data field. The contents returned in the Controller Health Data Structure for filtered fields are undefined.

Figure 81: Controller Health Status Changed Flags (CHSCF)

Bits	Reset	Description
15:13	0	Reserved
12	0	**Critical Warning (CWARN):** This bit is set to ‘1’ when any of the Critical Warning bits in the Controller Health Data Structure transition from ‘0’ to ‘1’.
11	0	**Available Spare (SPARE):** This bit is set to ‘1’ when the Available Spare field in the Controller Health Data Structure changes state.
10	0	**Percentage Used (PDLU):** This bit is set to ‘1’ when the Percentage Used field in the Controller Health Data Structure changes state.
09	0	**Composite Temperature Change (CTEMP):** This bit is set to ‘1’ when the Composite Temperature field in the Controller Health Data Structure changes state.
08	HwInit	**Controller Status Change (CSTS):** This bit is set to ‘1’ when the Shutdown Status field in the Controller Health Data Structure changes state or when the Ready, Controller Fatal Status, NVM Subsystem Reset Occurred, Controller Enable Change Occurred, Namespace Attribute Changed, or Firmware Activated bit in the Controller Health Data Structure transitions from ‘0’ to ‘1’.
07	HwInit	**Firmware Activated (FA):** This bit is set to ‘1’ when the Firmware Activated bit in the Controller Health Data Structure transitions from ‘0’ to ‘1’.
06	0	**Namespace Attribute Changed (NAC):** This bit is set to ‘1’ when the Namespace Attribute Changed bit in the Controller Health Data Structure transitions from ‘0’ to ‘1’.
05	0	**Controller Enable Change Occurred (CECO):** This bit is set to ‘1’ when the Controller Enable Change Occurred bit in the Controller Health Data Structure transitions from ‘0’ to ‘1’.
04	HwInit	**NVM Subsystem Reset Occurred (NSSRO):** This bit is set to ‘1’ when the NVM Subsystem Reset Occurred bit in the Controller Health Data Structure transitions from ‘0’ to ‘1’.
03	0	Reserved
02	0	**Shutdown Status (SHST):** This bit is set to ‘1’ when the Shutdown Status field in the Controller Health Data Structure changes state.
01	HwInit	**Controller Fatal Status (CFS):** This bit is set to ‘1’ when the Controller Fatal Status bit in the Controller Health Data Structure transitions from ‘0’ to ‘1’.
00	0	**Ready (RDY):** This bit is set to ‘1’ when the Ready bit in the Controller Health Data Structure transitions from ‘0’ to ‘1’.

Figure 82: Controller Health Data Structure to Controller Health Status Changed Flags Mapping

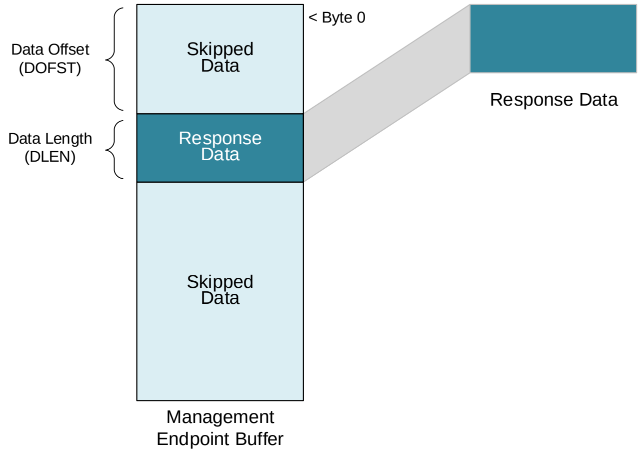


5.4 Management Endpoint Buffer Read

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 84 and Figure 85 respectively. There is no Request Data included in a Management Endpoint Buffer Read command. The NVMe Management Response field is reserved.

Figure 83: 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 Response with the PEL field indicating 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 Response with the PEL field indicating 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 Response Message Status of Management Endpoint Buffer Cleared Due to Sanitize.

Figure 84: Management Endpoint Buffer Read – NVMe Management Dword 0

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

Figure 85: Management Endpoint Buffer Read – NVMe Management Dword 1

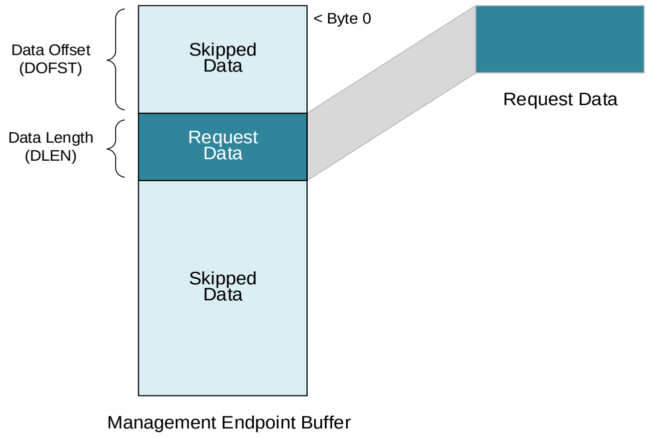
Bits	Description
31:16	Reserved
15:00	**Data Length (DLEN):** This field specifies the length, in bytes, to be transferred from the Management Endpoint Buffer starting at the byte offset specified by DOFST and returned in the Response Data. Specifying a value that is greater than the maximum supported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

5.5 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 87 and Figure 88 respectively. The NVMe Management Response field is reserved and there is no Response Data.

Figure 86: 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 Response with the PEL field indicating 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 Response with the PEL field indicating the DLEN field.

Figure 87: Management Endpoint Buffer Write – NVMe Management Dword 0

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

Figure 88: Management Endpoint Buffer Write – NVMe Management Dword 1

Bits	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 Response with the PEL field indicating this field. A Data Length value of 0h specifies that no data shall be transferred. This condition shall not be considered an error.

5.6 NVM Subsystem Health Status Poll

The NVM Subsystem Health Status Poll command is used to efficiently determine changes in health status attributes associated with the NVM Subsystem.

The NVM Subsystem Health Status Poll command operates independently using the out-of-band mechanism and the in-band tunneling mechanism.

An NVMe Storage Device or NVMe Enclosure supporting the NVM Subsystem Health Status Poll command using the out-of-band mechanism shall have an independent copy of the NVM Subsystem Health Data Structure (refer to Figure 90) dedicated to the out-of-band mechanism. In the out-of-band mechanism, an NVM Subsystem Health Status Poll command only applies to the copy of the NVM Subsystem Health Data Structure dedicated to the out-of-band mechanism.

An NVMe Storage Device or NVMe Enclosure supporting the NVM Subsystem Health Status Poll command using the in-band tunneling mechanism shall have an independent copy of the NVM Subsystem Health Data Structure dedicated to the in-band tunneling mechanism. In the in-band tunneling mechanism, an NVM Subsystem Health Status Poll command only applies to the copy of the NVM Subsystem Health Data Structure dedicated to the in-band tunneling mechanism.

The NVM Subsystem Health Status Poll command uses NVMe Management Dword 1 as shown in Figure 89.

Figure 89: NVM Subsystem Health Status Poll - NVMe Management Dword 1

Bits	Description
31	**Clear Status (CS):** When this bit is set to ‘1’, the state of reported Composite Controller Status is cleared.
30:00	Reserved

All other command specific fields are reserved.

The NVM Subsystem Health Data Structure, shown in Figure 90, is returned in the Response Data of a Successful Response Message. NVM Subsystem Health Status Poll Command responses do not use the NVMe Management Response field and this field is reserved. The Response Data field contains the NVM Subsystem Health Data Structure and is always the size of the NVM Subsystem Health Data Structure.

Figure 90: NVM Subsystem Health Data Structure (NSHDS)

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>0</td><td><b>NVM Subsystem Status (NSS):</b> This field indicates the status of the NVM Subsystem.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:6</td><td>Reserved</td></tr>
<tr><td>5</td><td><b>Drive Functional (DF):</b> This bit is set to ‘1’ to indicate an NVM Subsystem is functional. If cleared to ‘0’, then there is an unrecoverable failure detected in the NVM Subsystem.</td></tr>
<tr><td>4</td><td><b>Reset Not Required (RNR):</b> This bit is set to ‘1’ to indicate the NVM Subsystem does not require 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.</td></tr>
<tr><td>3</td><td><b>Port 0 PCIe Link Active (P0LA):</b> This bit 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.</td></tr>
<tr><td>2</td><td><b>Port 1 PCIe Link Active (P1LA):</b> This bit 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.</td></tr>
<tr><td>1:0</td><td>Reserved</td></tr>
</table></td></tr>
</table>

Figure 90: NVM Subsystem Health Data Structure (NSHDS)

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>1</td><td><b>Smart Warnings (SW):</b> This field contains the Critical Warning field (byte 0) of the NVMe SMART / Health Information log. Each bit in this field is inverted from the NVM Express Base Specification definition (i.e., the management interface shall indicate a ‘0’ value while the corresponding bit is set to ‘1’ in the log page). Refer to the NVM Express Base Specification for bit definitions.
If there are multiple Controllers in the NVM Subsystem, the Responder shall combine the Critical Warning field from every Controller in the NVM Subsystem such that a bit in this field is:
• Cleared to ‘0’ if any Controller in the NVM Subsystem indicates a critical warning for that corresponding bit; or
• Set to ‘1’ if all Controllers in the NVM Subsystem do not indicate a critical warning for the corresponding bit.</td></tr>
<tr><td>2</td><td><b>Composite Temperature (CTEMP):</b> This field indicates the current temperature in degrees Celsius. If a temperature value is reported, it should be the same temperature as the Composite Temperature from the SMART log of hottest Controller in the NVM Subsystem. The reported temperature range is vendor specific and shall not exceed the range -60 °C to +127 °C. The 8-bit format of the data is shown below.
This field should not report a temperature that is older than 1 s. If recent data is not available, the Responder should indicate a value of 80h for this field.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>00h to 7Eh</td><td>Temperature is measured in degrees Celsius (0 °C to 126 °C)</td></tr>
<tr><td>7Fh</td><td>127 °C or higher</td></tr>
<tr><td>80h</td><td>No temperature data or temperature data is more the 5 s old.</td></tr>
<tr><td>81h</td><td>Temperature sensor failure</td></tr>
<tr><td>82h to C3h</td><td>Reserved</td></tr>
<tr><td>C4h</td><td>Temperature is -60 °C or lower</td></tr>
<tr><td>C5h to FFh</td><td>Temperature measured in degrees Celsius is represented in two’s complement (-1 °C to -59 °C)</td></tr>
</table></td></tr>
<tr><td>3</td><td><b>Percentage Drive Life Used (PDLU):</b> Contains a vendor specific estimate of the percentage of NVM Subsystem NVM life used based on the actual usage and the manufacturer’s prediction of NVM life. If an NVM Subsystem has multiple Controllers, the highest value is returned. A value of 100 indicates that the estimated endurance of the NVM in the NVM Subsystem has been consumed but may not indicate an NVM Subsystem failure. The value is allowed to exceed 100. Percentages greater than 254 shall be represented as 255. This value should be updated once per power-on hour and equal the Percentage Used value in the NVMe SMART Health Log Page.</td></tr>
</table>

Figure 90: NVM Subsystem Health Data Structure (NSHDS)

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>5:4</td><td>

Composite Controller Status (CCS): This field reports the composite status of all Controllers in the NVM Subsystem.

The bits in this field are cleared after the NVM Subsystem Health Data Structure (refer to Figure 90) is returned in a Success Response associated with an NVM Subsystem Health Status Poll command where the Clear Status bit set. A Configuration Set command that selects Health Status Change may be used to clear selected bits to ‘0’.

<table>
<tr><th>Bits</th><th>Reset</th><th>Description</th></tr>
<tr><td>15:13</td><td>0</td><td>Reserved</td></tr>
<tr><td>12</td><td>0</td><td><b>Critical Warning (CWARN):</b> This bit is set to ‘1’ when the Critical Warning bit in the Controller Health Status Changed Flags transitions from ‘0’ to ‘1’ in one or more Controllers in the NVM Subsystem.</td></tr>
<tr><td>11</td><td>0</td><td><b>Available Spare (SPARE):</b> This bit is set to ‘1’ when the Available Spare bit in the Controller Health Status Changed Flags transitions from ‘0’ to ‘1’ in one or more Controllers in the NVM Subsystem.</td></tr>
<tr><td>10</td><td>0</td><td><b>Percentage Used (PDLU):</b> This bit is set to ‘1’ when the Percentage Used field in the Controller Health Status Changed Flags transitions from ‘0’ to ‘1’ in one or more Controllers in the NVM Subsystem.</td></tr>
<tr><td>09</td><td>0</td><td><b>Composite Temperature Change (CTEMP):</b> This bit is set to ‘1’ when the Composite Temperature field in the Controller Health Status Changed Flags transitions from ‘0’ to ‘1’ in one or more Controllers in the NVM Subsystem.</td></tr>
<tr><td>08</td><td>HwInit</td><td><b>Controller Status Change (CSTS):</b> This bit is set to ‘1’ when the Controller Status field in the Controller Health Status Changed Flags transitions from ‘0’ to ‘1’ in one or more Controllers in the NVM Subsystem.</td></tr>
<tr><td>07</td><td>HwInit</td><td><b>Firmware Activated (FA):</b> This bit is set to ‘1’ when the Firmware Activated bit in the Controller Health Status Changed Flags transitions from ‘0’ to ‘1’ in one or more Controllers in the NVM Subsystem.</td></tr>
<tr><td>06</td><td>0</td><td><b>Namespace Attribute Changed (NAC):</b> This bit is set to ‘1’ when the Namespace Attribute Changed bit in the Controller Health Status Changed Flags transitions from ‘0’ to ‘1’ in one or more Controllers in the NVM Subsystem.</td></tr>
<tr><td>05</td><td>0</td><td><b>Controller Enable Change Occurred (CECO):</b> This bit is set to ‘1’ when the Controller Enable Change Occurred bit in the Controller Health Status Changed Flags transitions from ‘0’ to ‘1’ in one or more Controllers in the NVM Subsystem.</td></tr>
<tr><td>04</td><td>HwInit</td><td><b>NVM Subsystem Reset Occurred (NSSRO):</b> This bit is set to ‘1’ when the value of the NVM Subsystem Reset Occurred (CSTS.NSSRO) bit transitions from a ‘0’ to a ‘1’ in one or more Controllers in the NVM Subsystem.</td></tr>
<tr><td>03</td><td>0</td><td>Reserved</td></tr>
<tr><td>02</td><td>0</td><td><b>Shutdown Status (SHST):</b> This bit is set to ‘1’ when the Shutdown Status bit in the Controller Health Status Changed Flags transitions from ‘0’ to ‘1’ in one or more Controllers in the NVM Subsystem.</td></tr>
<tr><td>01</td><td>HwInit</td><td><b>Controller Fatal Status (CFS):</b> This bit is set to ‘1’ when the Controller Fatal Status bit in the Controller Health Status Changed Flags transitions from ‘0’ to ‘1’ in one or more Controllers in the NVM Subsystem.</td></tr>
<tr><td>00</td><td>0</td><td><b>Ready (RDY):</b> This bit is set to ‘1’ when the Ready bit in the Controller Health Status Changed Flags transitions from ‘0’ to ‘1’ in one or more Controllers in the NVM Subsystem.</td></tr>
</table>

</td></tr>
<tr><td>7:6</td><td>Reserved</td></tr>
</table>

5.7 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 and Dword 1. The format of NVMe Management Dword 0 is shown in Figure 91 and the format of NVMe Management Dword 1 is shown in Figure 92. There is no Request Data included in a Read NVMe-MI Data Structure command.

Some port-specific Data Structure Types are accessible from any Responder. Other port-specific Data Structure Types (e.g., Optionally Supported Command List) are only accessible from the Responder that received the Command Message.

Figure 91: Read NVMe-MI Data Structure – NVMe Management Dword 0

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:24</td><td><b>Data Structure Type (DTYP):</b> 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>Optionally Supported Command List</td></tr>
<tr><td>05h</td><td>Management Endpoint Buffer Command Support List</td></tr>
<tr><td>06h to FFh</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>23:16</td><td><b>Port Identifier (PORTID):</b> This field contains the identifier of the port whose data structure is returned.
If the DTYP field value is 01h (Port Information) or 05h (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><b>Controller Identifier (CTRLID):</b> This field contains the Controller Identifier whose data structure is returned.
If the DTYP field value is 02h (Controller List), 03h (Controller Information), or 04h (Optionally Supported Command List), then this field contains the Controller Identifier in the NVM Subsystem whose information is requested.
If the DTYP field value is 04h (Optionally Supported Command List), then this field is only applicable for commands in the Optionally Supported Command List Data Structure with NMIMT set to a value of 02h (NVMe Admin Command) and shall be ignored for commands with NMIMT set to any value other than 02h.
For all other values of the DTYP field, this field is reserved.</td></tr>
</table>

Figure 92: Read NVMe-MI Data Structure – NVMe Management Dword 1

Bits	Description
31:08	Reserved
07:00	**I/O Command Set Identifier (IOCSI):** If the DTYP field value corresponds to Optionally Supported Command List or the Management Endpoint Buffer Command Support List, then this field specifies the I/O Command Set used to select the optional I/O Command Set Specific Admin commands. For more information about I/O Command Sets refer to the NVM Express Base Specification. For all other values of the DTYP field, this field is reserved.

Upon successful completion of the Read NVMe-MI Data Structure, the NVMe Management Response field is shown in Figure 93 and the specified data structure is returned in the Response Data.

Figure 93: Read NVMe-MI Data Structure – NVMe Management Response

Bits	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 94.

Figure 94: NVM Subsystem Information Data Structure

Bytes	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 set to 2h 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 95.

Figure 95: Port Information Data Structure

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>00</td><td><b>Port Type:</b> Specifies the port type.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0h</td><td>Inactive</td></tr>
<tr><td>1h</td><td>PCIe</td></tr>
<tr><td>2h</td><td>SMBus</td></tr>
<tr><td>3h to FFh</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>01</td><td><b>Port Capabilities:</b> This field contains information about the capabilities of the port.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:1</td><td>Reserved</td></tr>
<tr><td>0</td><td><b>Command Initiated Auto Pause Supported (CIAPS):</b> If this bit is set to '1', then the Command Initiated Auto Pause (CIAP) bit is supported in Command Messages on this port. If this bit is cleared to '0', then the CIAP bit is not supported in Command Messages on this port.</td></tr>
</table></td></tr>
<tr><td>03:02</td><td><b>Maximum MCTP Transmission Unit Size:</b> 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 cleared to 0h.
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 (refer to the PCI Express Base Specification), 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.</td></tr>
<tr><td>07:04</td><td><b>Management Endpoint Buffer Size:</b> This field specifies the size of the Management Endpoint Buffer in bytes when a Management Endpoint Buffer is supported.
A value of 0h in this field indicates that the Management Endpoint does not support a Management Endpoint Buffer.</td></tr>
<tr><td>31:08</td><td>Port Type Specific (refer to Figure 96 and Figure 97)</td></tr>
</table>

Figure 96: PCIe Port Specific Data

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>08</td><td><b>PCIe Maximum Payload Size:</b> 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>1 KiB</td></tr>
<tr><td>4h</td><td>2 KiB</td></tr>
<tr><td>5h</td><td>4 KiB</td></tr>
<tr><td>6h to FFh</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>09</td><td><b>PCIe Supported Link Speeds Vector:</b> This field indicates the Supported Link Speeds for the specified PCIe port.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:5</td><td>Reserved</td></tr>
<tr><td>4</td><td>Set to ‘1’ if the PCIe link supports 32.0 GT/s, otherwise cleared to ‘0’.</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>10</td><td><b>PCIe Current Link Speed:</b> 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 to FFh</td><td>Reserved</td></tr>
</table></td></tr>
</table>

Figure 96: PCIe Port Specific Data

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>11</td><td><b>PCIe Maximum Link Width:</b> 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 Requester 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 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>12</td><td><b>PCIe Negotiated Link Width:</b> 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 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>13</td><td><b>PCIe Port Number:</b> 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 (refer to the NVMe over PCIe Transport Specification).</td></tr>
<tr><td>31:14</td><td>Reserved</td></tr>
</table>

Figure 97: SMBus Port Specific Data

Bytes	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.

Figure 97: SMBus Port Specific Data

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>09</td><td><b>Maximum VPD Access SMBus/I2C Frequency:</b> 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>4h to FFh</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>10</td><td><b>Current Management Endpoint SMBus/I2C Address:</b> This field indicates the current MCTP SMBus/I2C address. A value of 0h indicates there is no Management Endpoint on this port.</td></tr>
<tr><td>11</td><td><b>Maximum Management Endpoint SMBus/I2C Frequency:</b> 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>4h to FFh</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>12</td><td><b>NVMe Basic Management</b>
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:1</td><td>Reserved</td></tr>
<tr><td>0</td><td>If set to ‘1’, then the port implements the NVMe Basic Management Command. If cleared to ‘0’, then the port does not implement the NVMe Basic Management Command.
It is strongly recommended that implementations clear this bit to ‘0’. The NVMe Basic Management Command is included in Appendix A for information purposes only and is not a part of the standard NVMe-MI protocol.</td></tr>
</table></td></tr>
<tr><td>31:13</td><td>Reserved</td></tr>
</table>

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 2,047 Controller identifiers. Refer to the NVM Express Base Specification for a definition of the Controller List.

Figure 98: Controller Information Data Structure

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>00</td><td><b>Port Identifier (PORTID):</b> 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><b>PCIe Routing ID Information (PRII):</b> This field provides additional data about the PCI Express Routing ID (PRI) for the specified Controller.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:1</td><td>Reserved</td></tr>
<tr><td>0</td><td><b>PCIe Routing ID Valid:</b> 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 cleared to ‘0’ if the device has not captured a Bus and Device number (Bus Number only for ARI devices).</td></tr>
</table></td></tr>
</table>

Figure 98: Controller Information Data Structure

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>07:06</td><td><b>PCIe Routing ID (PRI):</b> This field contains the PCIe Routing ID for the specified Controller.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>15:08</td><td><b>PCI Bus Number:</b> The Controller's PCI Bus Number.</td></tr>
<tr><td>07:03</td><td><b>PCI Device Number:</b> The Controller's PCI Device Number.</td></tr>
<tr><td>02:00</td><td><b>PCI Function Number:</b> 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><b>PCI Vendor ID:</b> The PCI Vendor ID for the specified Controller.</td></tr>
<tr><td>11:10</td><td><b>PCI Device ID:</b> The PCI Device ID for the specified Controller.</td></tr>
<tr><td>13:12</td><td><b>PCI Subsystem Vendor ID:</b> The PCI Subsystem Vendor ID for the specified Controller.</td></tr>
<tr><td>15:14</td><td><b>PCI Subsystem Device ID:</b> 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 Responder supports. The I/O Command Set Identifier (IOCSI) field in NVMe Management Dword 1 selects the I/O Command Set for the I/O Command Set Specific Admin commands that are returned in the Optionally Supported Command List data structure. The Optionally Supported Command List data structure may contain up to 2,047 commands, and shall be minimally sized (i.e., if there is one optionally supported command, the data structure is 4 bytes total).

Figure 99: Optionally Supported Command List Data Structure

Bytes	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 100.
05:04	**Command 1 (CMD1):** This field contains the Command Type and Opcode for the second optionally supported command, if applicable. Refer to Figure 100.
…	
(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 100.

Figure 100: Optionally Supported Command Data Structure

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>00</td><td><b>Command Type:</b> This field specifies the type of command used by the optionally supported command.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7</td><td>Reserved</td></tr>
<tr><td>6:3</td><td><b>NVMe-MI Message Type (NMIMT):</b> This field specifies the NVMe-MI Message Type. Refer to Figure 19.</td></tr>
<tr><td>2:0</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>01</td><td><b>Opcode:</b> This field specifies the opcode used for the optionally supported command.</td></tr>
</table>

If the Management Endpoint Buffer Size field in the Port Information Data Structure is not 0h, 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 0h, 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 I/O Command Set Identifier (IOCSI) field in NVMe Management Dword 1 selects the I/O Command Set for the I/O Command Set Specific Admin commands that are returned in the Management Endpoint Buffer Command Support List data structure. The data structure may contain up to 2,047 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 101: Management Endpoint Buffer Supported Command List Data Structure

Bytes	Description
01:00	**Number of Commands (NUMCMD):** This field contains the number of 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 Management Endpoint Buffer Supported Command Data Structure (refer to Figure 102) 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 102) 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 102) for the N+1 command that supports the use of the Management Endpoint Buffer associated with the Management Endpoint.

Figure 102: Management Endpoint Buffer Supported Command Data Structure

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>00</td><td><b>Command Type:</b> This field specifies the type of command 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><b>NVMe-MI Message Type (NMIMT):</b> This field specifies the NVMe-MI Message Type. Refer to Figure 19.</td></tr>
<tr><td>2:0</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>01</td><td><b>Opcode:</b> This field specifies the opcode of the command that supports the Management Endpoint Buffer.</td></tr>
</table>

5.8 Reset

The Reset command may be used to initiate a reset.

The Reset command uses NVMe Management Dword 0. The format of NVMe Management Dword 0 is shown in Figure 103. All other command specific fields in the Request Message and Response Message are reserved.

Figure 103: Reset - NVMe Management Dword 0

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:24</td><td><b>Reset Type:</b> This field specifies the type of reset to be performed.
<table>
<tr><th>Value</th><th>O/M[1]</th><th>Description</th></tr>
<tr><td>00h</td><td>M[2]</td><td>Reset NVM Subsystem</td></tr>
<tr><td>01h to FFh</td><td>-</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>23:00</td><td>Reserved</td></tr>
<tr><td colspan="2">NOTES:
1. O/M definition: O = Optional, M = Mandatory
2. Required if the NVM Subsystem Reset feature is supported via the NSSR property as defined in the NVM Express Base Specification; else, it is optional.</td></tr>
</table>

When a Reset command that specifies a Reset NVM Subsystem in the Reset Type field completes successfully, the NVM Subsystem Reset is initiated (refer to section 8.3). No Success Response is transmitted.

A Management Controller should shutdown all NVMe Controllers in an NVM Subsystem prior to resetting the NVM Subsystem. Refer to the Shutdown command in section 5.11.

5.9 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 104 and the format of NVMe Management Dword 1 is shown in Figure 105. 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 an SES control type diagnostic page, then the Responder responds with an Invalid Parameter Error Response with the PEL field indicating the PCODE field.

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 an SES Receive command, then the Responder responds with an Invalid Parameter Error Response with the PEL field indicating the invalid field. If a condition occurs that in SES-3 results in a CHECK CONDITION, then the Responder responds with an Error Response. The mapping of Error Response Status values to SES-3 sense keys and additional sense codes is shown in Figure 13.

If the SES Receive command is supported in the out-of-band mechanism, then the Management Endpoint Buffer 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 Requester 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 104: SES Receive – NVMe Management Dword 0

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

Figure 105: SES Receive – NVMe Management Dword 1

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

Figure 106: SES Receive – NVMe Management Response

Bits	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.

5.10 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 (PCODE) field 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 Responder responds with an Invalid Parameter Error Response with the PEL field indicating the PCODE field.

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 Responder responds with an Invalid Parameter Error Response with the PEL field indicating the invalid field. If a condition occurs that in SES-3 results in a CHECK CONDITION, then the Responder responds with an Error Response. The mapping of Response Message Status values to SES-3 sense keys and additional sense codes is shown in Figure 13.

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 0h and no data is valid, and results in a Success Response. 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 Responder 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 Requester to transfer an SES control type diagnostic page whose size exceeds the maximum size allowed by one NVMe-MI Message.

Figure 107: SES Send – NVMe Management Dword 1

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

5.11 Shutdown

The Shutdown command sent to one Management Endpoint initiates a shutdown on all Controllers in the NVM Subsystem.

The Shutdown command uses NVMe Management Dword 0. The format of NVMe Management Dword 0 is shown in Figure 108. All other command specific fields in the Request Message and Response Message are reserved.

Figure 108: Shutdown - NVMe Management Dword 0

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>31:24</td><td><b>Shutdown Type:</b> This field specifies the type of shutdown to be performed.
<table>
<tr><th>Value</th><th>O/M[1]</th><th>Description</th></tr>
<tr><td>00h</td><td>O[2]</td><td>Normal NVM Subsystem Shutdown</td></tr>
<tr><td>01h</td><td>O[2]</td><td>Abrupt NVM Subsystem Shutdown</td></tr>
<tr><td>02h to FFh</td><td>-</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>23:00</td><td>Reserved</td></tr>
<tr><td colspan="2">NOTES:
1. O/M definition: O = Optional, M = Mandatory
2. Mandatory for the out-of-band mechanism if the NVM Subsystem Shutdown feature is supported on all NVMe Controllers in the NVM Subsystem (refer to the NVM Express Base Specification).</td></tr>
</table>

Upon receipt of a Shutdown command specifying a Normal NVM Subsystem Shutdown, then: for each Controller in the NVM Subsystem:

- if:
  - CSTS.SHST is cleared to 00b on that Controller; and
  - An outstanding Asynchronous Event Request command exists on that Controller (refer to the NVM Express Base Specification),

then the Controller shall issue a Normal NVM Subsystem Shutdown event prior to shutting down the Controller (refer to the NVM Express Base Specification);

- a normal shutdown is initiated on the Controller as specified by the NVM Express Base Specification.

Upon receipt of a Shutdown command specifying an Abrupt NVM Subsystem Shutdown, then for each Controller in the NVM Subsystem an abrupt shutdown is initiated as specified by the NVM Express Base Specification.

The Shutdown command completes successfully when all NVMe Controllers in the NVM Subsystem report shutdown process complete (i.e., CSTS.SHST is set to 10b and CSTS.ST is set to '1'). Refer to the NVMe Express Base Specification on the condition when it is safe to power down the NVM Subsystem.

5.12 VPD Read

The VPD Read command is used to read the Vital Product Data described in section 8.2. Upon successful completion of the VPD Read command, the specified portion of the VPD contents is returned in the Response Data.

The VPD Read command uses NVMe Management Dword 0 and 1. The format of NVMe Management Dwords 0 and 1 are shown in Figure 109 and Figure 110 respectively. There is no Request Data sent in the Request Message.

A VPD Read command with length 0 and no data is valid. The Responder responds with a Success Response and no Response Data. If the Data Length plus Data Offset fields are greater than the size of the VPD, then the Responder does not return the VPD contents and responds with an Invalid Parameter Error Response with the PEL field indicating the Data Length field.

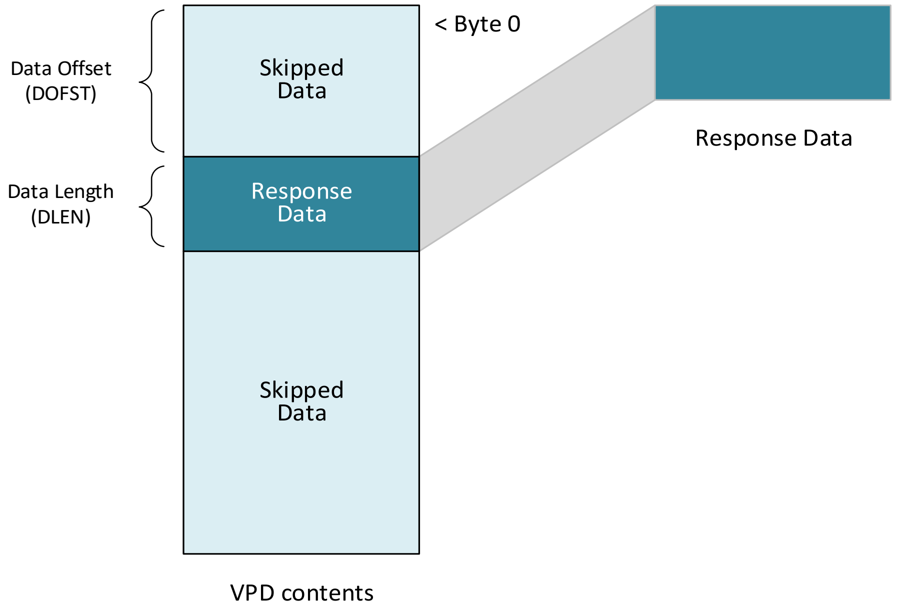
Figure 109: VPD Read NVMe Management Dword 0

Bits	Description
31:16	Reserved
15:00	**Data Offset (DOFST):** This field specifies the starting offset, in bytes, into the VPD data that is contained in the Response Message.

Figure 110: VPD Read NVMe Management Dword 1

Bits	Description
31:16	Reserved
15:00	**Data Length (DLEN):** This field specifies the length, in bytes, to be read from the VPD starting at the byte offset specified by DOFST.

Figure 111: VPD Read Response Data



5.13 VPD Write

The VPD Write command is used to update the Vital Product Data described in section 8.2.

After the VPD Write command has been processed without error, reading the contents of the FRU Information Device directly or a VPD Read command processed without error shall return the new VPD contents (i.e., those supplied with the VPD Write command). The data to be written to the VPD is specified in the Request Data field. VPD Write uses NVMe Management Dwords 0 and 1 as shown in Figure 112 and Figure 113.

The VPD contents should be capable of being updated at least 8 times using the VPD Write command[1]. If the initial value of the VPD Write Cycles Remaining field is less than 100, then the VPD Write Cycle Remaining Valid bit should be set to '1' (Refer to the VPD Write Cycle Information field in the Identify Controller data structure of the NVM Express Base Specification). If there is an error preventing update of the VPD contents, then the Responder responds with a Generic Error Response and VPD Writes Exceeded status.

A VPD Write command with Data Length 0h and no data is valid. The Responder responds with a Success Response.

Figure 112: VPD Write – NVMe Management Dword 0

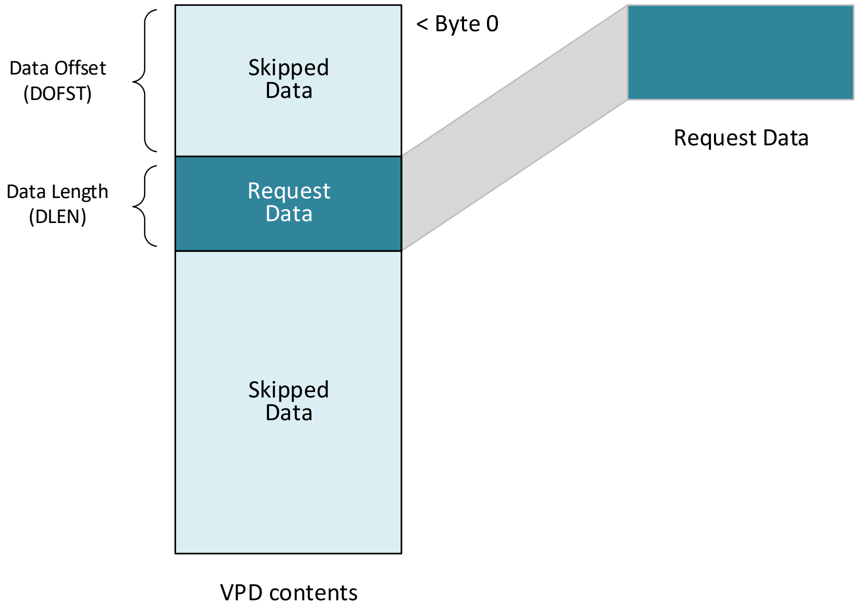
Bits	Description
31:16	Reserved
15:00	**Data Offset (DOFST):** This field specifies the starting offset, in bytes, into the VPD data that is written.

[1] NVM Express Management Interface Specification, Revision 1.0a and prior recommended that VPD contents should be capable of being updated at least 100 times using the VPD Write command.

Figure 113: VPD Write – NVMe Management Dword 1

Bits	Description
31:16	Reserved
15:00	**Data Length (DLEN):** This field specifies the length, in bytes, to be written to the VPD starting at the byte offset specified by DOFST.

Figure 114: VPD Write Request Data



The Requester should not read the contents of the VPD while this command is servicing. Reading the contents of the VPD or the processing of a VPD Read command while a VPD Write command is being processed may return incorrect data as a result of the read.

If the Data Length plus Data Offset fields are greater than the size of the VPD, then the Responder does not write to the VPD and responds with an Invalid Parameter Error Response with the PEL field indicating the Data Length field.

6 NVM Express Admin Command Set

The NVM Express Admin Command Set allows NVMe Admin Commands to be issued to any Controller in the NVM Subsystem using the out-of-band mechanism. Figure 115 shows NVMe Admin Commands that are mandatory, optional, and prohibited for an NVMe Storage Device and an NVMe Enclosure using the out-of-band mechanism. All NVMe Admin Commands are prohibited using the in-band tunneling mechanism. The commands are defined in the NVM Express Base Specification and the I/O Command Set specifications. If an NVMe Admin Command is issued in a Request Message that is a prohibited command in Figure 115, the Management Endpoint shall return an Invalid Parameter Error Response with PEL field indicating the NVMe opcode. Future revisions of this specification may add additional commands to Figure 115. The NVM Express Admin Command Set is only applicable in the out-of-band mechanism and is prohibited in the in-band tunneling mechanism.

Figure 115: List of NVMe Admin Commands Supported using the Out-of-Band Mechanism

Command	Opcode	NVMe Storage Device O/M/P[1]	NVMe Enclosure O/M/P[1]	Reference Specification
Abort	00h	P	P	NVMe Base Specification
Asynchronous Event Request	0Ch	P	P	NVMe Base Specification
Capacity Management	20h	O	P	NVMe Base Specification
Create I/O Completion Queue	05h	P	P	NVMe Base Specification
Create I/O Submission Queue	01h	P	P	NVMe Base Specification
Delete I/O Completion Queue	04h	P	P	NVMe Base Specification
Delete I/O Submission Queue	00h	P	P	NVMe Base Specification
Device Self-test	14h	O	O	NVMe Base Specification
Directive Receive	1Ah	P	P	NVMe Base Specification
Directive Send	19h	P	P	NVMe Base Specification
Doorbell Buffer Config	7Ch	P	P	NVMe Base Specification
Firmware Commit	10h	O	O	NVMe Base Specification
Firmware Image Download	11h	O	O	NVMe Base Specification
Format NVM	80h	O	P	NVMe Base Specification
Get Features	0Ah	M	O	NVMe Base Specification
Get LBA Status	86h	O	P	NVM Command Set Specification
Get Log Page[2]	02h	M	O	NVMe Base Specification
Identify	06h	M	O	NVMe Base Specification
Keep Alive	18h	P	P	NVMe Base Specification
Lockdown	24h	O	O	NVMe Base Specification
Namespace Management	0Dh	O	P	NVMe Base Specification
Namespace Attachment	15h	O	P	NVMe Base Specification
NVMe-MI Receive	1Dh	P	P	NVMe Base Specification
NVMe-MI Send	1Eh	P	P	NVMe Base Specification
Sanitize	84h	O	O	NVMe Base Specification
Security Send	81h	O	P	NVMe Base Specification
Security Receive	82h	O	P	NVMe Base Specification
Set Features	09h	O	O	NVMe Base Specification
Vendor Specific	C0h to FFh	O	O	NVMe Base Specification

Figure 115: List of NVMe Admin Commands Supported using the Out-of-Band Mechanism

Command	Opcode	NVMe Storage Device O/M/P [1]	NVMe Enclosure O/M/P [1]	Reference Specification
Virtualization Management	1Ch	O	O	NVMe Base Specification
Fabrics Commands	7Fh	P	P	NVMe Base Specification

NOTES:

1. O/M/P 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 or an NVMe Enclosure and may implement optional commands allowed by either an NVMe Storage Device or an NVMe Enclosure. Mandatory commands shall be supported using the out-of-band mechanism if the NVMe Controller specified by the Controller ID field supports the command in-band.
2. If the Retain Asynchronous Event bit is cleared to '0', then the status associated with the NVMe Admin Command shall be Invalid Field in Command (i.e., the NVMe Admin command is aborted). For implementations compliant to version 1.1 or earlier of this specification, the Retain Asynchronous Event bit in the Get Log Page command (refer to the NVM Express Base Specification) may or may not be ignored by the Controller. Refer to section 6.2.

NVMe Admin Commands over the out-of-band mechanism may interfere with host software. A Management Controller should coordinate with the host or issue only NVMe Admin Commands that do not interfere with host software or in band NVMe commands (e.g., Identify). Coordination between a Management Controller and host is outside the scope of this specification.

NVMe Admin Commands over the out-of-band mechanism may target a Controller that is disabled or held in reset by the host. When this occurs, the NVMe Admin Command is processed normally.

The Request Message format for NVMe Admin Commands is shown in Figure 116 and is described Figure 117.

Figure 116: NVMe Admin Command Request Format

<table>
<tr><th colspan="8">+3</th><th colspan="8">+2</th><th colspan="8">+1</th><th colspan="8">+0</th><th></th></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td></td></tr>
<tr><td colspan="14">Reserved</td><td>CIAP</td><td>MEB</td><td>ROR</td><td colspan="4">NVMe-MI Msg Type</td><td colspan="2">R</td><td>CSI</td><td>IC</td><td colspan="7">Message Type</td><td>< Byte 0</td></tr>
<tr><td colspan="16">Controller ID</td><td colspan="8">Command Flags</td><td colspan="8">Opcode</td><td>< Byte 4</td></tr>
<tr><td colspan="32">Submission Queue Entry Dword 1</td><td>< Byte 8</td></tr>
<tr><td colspan="32">…</td><td></td></tr>
<tr><td colspan="32">Submission Queue Entry Dword 5</td><td>< Byte 24</td></tr>
<tr><td colspan="32">Data Offset</td><td>< Byte 28</td></tr>
<tr><td colspan="32">Data Length</td><td>< Byte 32</td></tr>
<tr><td colspan="32">Reserved</td><td>< Byte 36</td></tr>
<tr><td colspan="32">Reserved</td><td>< Byte 40</td></tr>
<tr><td colspan="32">Submission Queue Entry Dword 10</td><td>< Byte 44</td></tr>
<tr><td colspan="32">…</td><td></td></tr>
<tr><td colspan="32">Submission Queue Entry Dword 15</td><td>< Byte 64</td></tr>
<tr><td colspan="32">NVMe Request Data (optional)</td><td>< Bytes 68 to N-1</td></tr>
<tr><td colspan="32">Message Integrity Check</td><td>< Byte N</td></tr>
</table>

Figure 117: NVMe Admin Command Request Description

<table>
<tr><th>Bytes</th><th colspan="2">Description</th></tr>
<tr><td>03:00</td><td colspan="2"><b>NVMe-MI Message Header:</b> Refer to section 3.1.</td></tr>
<tr><td>04</td><td colspan="2"><b>Opcode (OPC):</b> This field specifies the opcode of the command. Refer to the NVM Express Base Specification.</td></tr>
<tr><td rowspan="5">05</td><td colspan="2"><b>Command Flags (CFLGS):</b> This field specifies flags for the command.</td></tr>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:2</td><td>Reserved</td></tr>
<tr><td>1</td><td><b>DOFST Valid (DOFSTV):</b> This bit is not used and shall be ignored by the Management Endpoint for implementations compliant with versions of this specification later than 1.1.</td></tr>
<tr><td>0</td><td><b>DLEN Valid (DLENV):</b> This bit is not used and shall be ignored by the Management Endpoint for implementations compliant with versions of this specification later than 1.1.</td></tr>
<tr><td>07:06</td><td colspan="2"><b>Controller ID (CTLID):</b> This field specifies the Controller ID of the Controller that this command targets.</td></tr>
</table>

Figure 117: NVMe Admin Command Request Description

Bytes	Description
11:08	**Submission Queue Entry Dword 1 (SQEDW1):** Submission Queue Entry Dword 1 as defined in the NVM Express Base Specification.
15:12	**Submission Queue Entry Dword 2 (SQEDW2):** Submission Queue Entry Dword 2 as defined in the NVM Express Base Specification.
19:16	**Submission Queue Entry Dword 3 (SQEDW3):** Submission Queue Entry Dword 3 as defined in the NVM Express Base Specification.
23:20	**Submission Queue Entry Dword 4 (SQEDW4):** Submission Queue Entry Dword 4 as defined in the NVM Express Base Specification.
27:24	**Submission Queue Entry Dword 5 (SQEDW5):** Submission Queue Entry Dword 5 as defined in the NVM Express Base Specification.
31:28	**Data Offset (DOFST):** For commands that transmit data from the Management Controller to the Management Endpoint (i.e., the Data Transfer subfield for the opcode field of the NVMe Admin Command as defined by the NVM Express Base Specification is 01b) or do not transmit data (i.e., the Data Transfer subfield in the opcode field of the NVMe Admin Command as defined by the NVM Express Base Specification is 00b), this field should be cleared to 0h. If this field is not 0h for commands that transmit data from the Management Controller to the Management Endpoint or that do not transfer data, then the Management Endpoint shall respond with an Invalid Parameter Error Response with the PEL field indicating this field. For commands that transmit data from the Management Endpoint to the Management Controller (i.e., the Response Data field in the Response Message has non-zero length), this field specifies the starting offset, in bytes, of the portion of data contained in the NVMe Admin Command completion data that is returned starting at byte offset 0h of the Response Data field in the Response Message. This field should be less than the size of the NVMe Admin Command completion data. If this field is greater than or equal to the size of the NVMe Admin Command completion data, then the Management Endpoint shall respond with an Invalid Parameter Error Response with the PEL field indicating this field. Bits 1:0 of this field should be cleared to 00b. If bits 1:0 are not cleared to 00b, then the Management Endpoint should respond with an Invalid Parameter Error Response with the PEL field indicating this field.

Figure 117: NVMe Admin Command Request Description

Bytes	Description
35:32	**Data Length (DLEN):** For commands that do not transfer data (i.e., the Data Transfer subfield in the opcode field of the NVMe Admin Command as defined by the NVM Express Base Specification is 00b), this field should be cleared to 0h. If this field is not 0h for commands that do not transfer data, then the Management Endpoint shall respond with an Invalid Parameter Error Response with the PEL field indicating this field. For commands that transmit data from the Management Controller to the Management Endpoint (i.e., the Data Transfer subfield in the opcode field of the NVMe Admin Command as defined by the NVM Express Base Specification is 01b), this field specifies the length, in bytes, of the data contained in the Request Data field in the Request Message. For commands that transmit data from the Management Endpoint to the Management Controller (i.e., the Data Transfer subfield in the opcode field of the NVMe Admin Command as defined by the NVM Express Base Specification is 10b), this field specifies the length, in bytes, of the portion of data contained in the NVMe Admin Command completion data that is returned in the Response Data field in the Response Message. The sum of DLEN plus DOFST should be less than or equal to the size of the NVMe Admin Command completion data. If the sum is greater, then the Management Endpoint shall respond with an Invalid Parameter Error Response with the PEL field indicating this field. For commands that transmit data (i.e., the Data Transfer subfield in the opcode of the NVMe Admin Command as defined by the NVM Express Base Specification is 01b or 10b), the Management Controller should specify a non-zero length in this field. If this field is cleared to 0h for commands that transmit data, then the Management Endpoint should respond with an Invalid Parameter Error Response with the PEL field indicating this field. Bits 1:0 of this field should be cleared to 00b. If bits 1:0 are not cleared to 00b, then the Management Endpoint shall respond with an Invalid Parameter Error Response with the PEL field indicating this field. This field should be less than or equal to 4,096. If this field is greater than 4,096, then the Management Endpoint shall respond with an Invalid Parameter Error Response with the PEL field indicating this field.
43:36	Reserved
47:44	**Submission Queue Entry Dword 10 (SQEDW10):** Submission Queue Entry Dword 10 as defined in the NVM Express Base Specification.
51:48	**Submission Queue Entry Dword 11 (SQEDW11):** Submission Queue Entry Dword 11 as defined in the NVM Express Base Specification.
55:52	**Submission Queue Entry Dword 12 (SQEDW12):** Submission Queue Entry Dword 12 as defined in the NVM Express Base Specification.
59:56	**Submission Queue Entry Dword 13 (SQEDW13):** Submission Queue Entry Dword 13 as defined in the NVM Express Base Specification.
63:60	**Submission Queue Entry Dword 14 (SQEDW14):** Submission Queue Entry Dword 14 as defined in the NVM Express Base Specification.
67:64	**Submission Queue Entry Dword 15 (SQEDW15):** Submission Queue Entry Dword 15 as defined in the NVM Express Base Specification.
N-1:68	NVMe Request Data (Optional)
N+3:N	**Message Integrity Check (MIC):** Refer to section 3.1.

The Response Message contains the corresponding format for NVMe Admin Commands is shown in Figure 118 and is described in Figure 119.

Figure 118: NVMe Admin Command Response Format

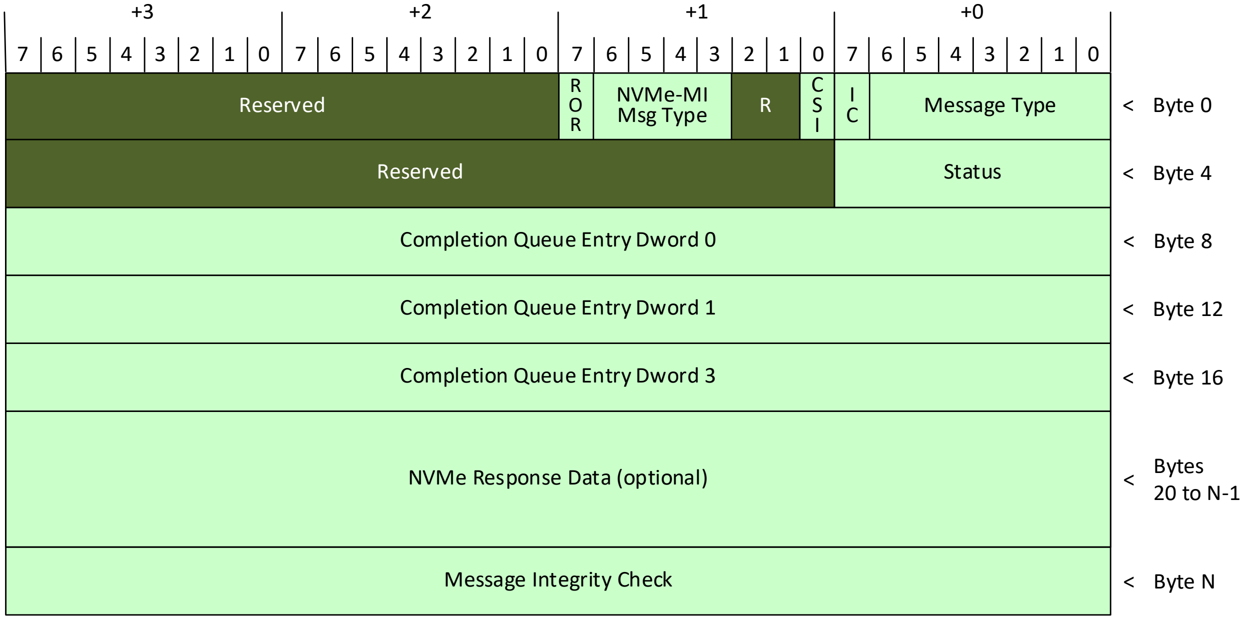


Figure 119: NVMe Admin Command Response Description

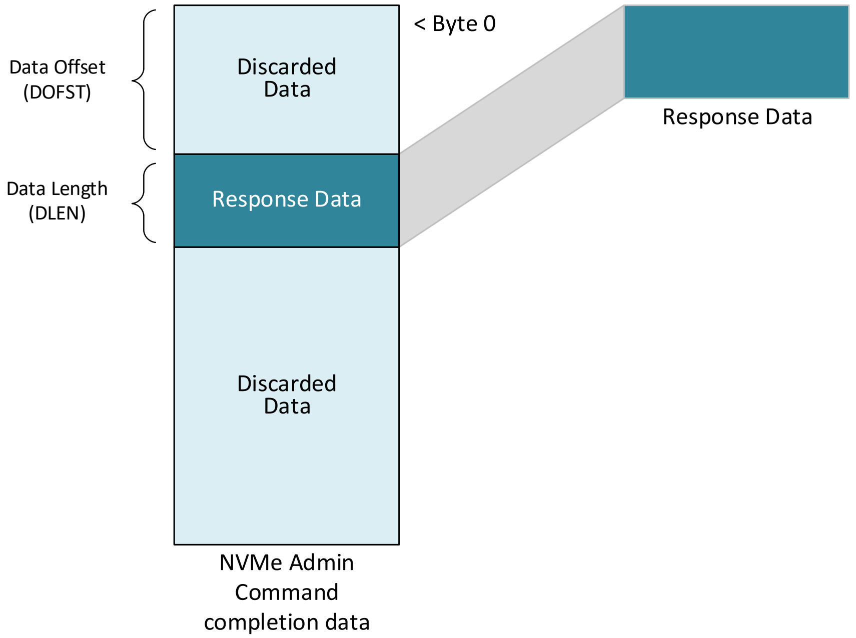
Bytes	Description
03:00	**NVMe-MI Message Header:** Refer to section 3.1.
04	**Status:** This field indicates the status of the NVMe Admin command. Refer to section 4.1.2.
07:05	Reserved
11:08	**Completion Queue Entry Dword 0 (CQEDW0):** Completion Queue Entry Dword 0 as defined in the NVM Express Base Specification.
15:12	**Completion Queue Entry Dword 1 (CQEDW1):** Completion Queue Entry Dword 1 as defined in the NVM Express Base Specification.
19:16	**Completion Queue Entry Dword 3 (CQEDW3):** Completion Queue Entry Dword 3 as defined in the NVM Express Base Specification. The Command ID field shall be cleared to 0h.
N-1:20	NVMe Response Data (Optional)
N+3:N	**Message Integrity Check:** Refer to section 3.1.

6.1 Request and Response Data

NVMe Admin Commands may contain data as part of the Command Message. This data is passed in the Request Data field instead of using PRP Lists or SGL segments. The PRP Entry 2 (PRP2) and Metadata Pointer (MPTR) fields within the NVMe Admin Commands are reserved.

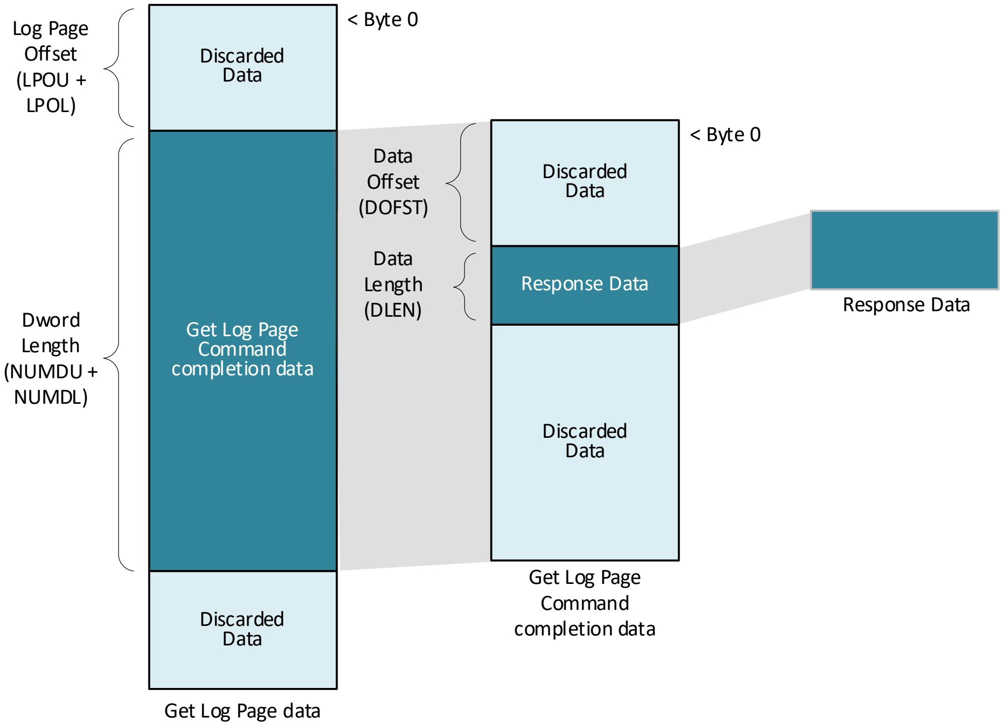
If there is Response Data expected in the Response Message in the completion of the NVMe Admin Command (i.e., the Data Transfer subfield in the corresponding NVMe Admin Command for the opcode is 10b), then the Data Offset and Data Length fields describe the portion of the NVMe Admin Command completion data that is transferred in the Response Message. Any remaining data not transferred in the Response Message is discarded by the Management Endpoint as shown in Figure 120.

Figure 120: NVMe Admin Command Response Data Example



Some NVMe Admin Commands specify an offset and length which shall be applied first to create the NVMe Admin Command completion data. Then DOFST and DLEN shall be applied to that Admin Command completion data which may further reduce the amount of Response Data. Figure 121 provides an example for the Get Log Page command.

Figure 121: NVMe Get Log Page Command Response Data Example



6.2 Status

A Response Message for an NVMe Admin Command may contain two status fields. The first status field, contained in Byte 4 of the Response Message, is defined by this specification, and the second Status Field, if present, is contained in Completion Queue Entry Dword 3 and defined in the NVM Express Base Specification.

An NVMe Admin Command Request Message is well formed if it does not contain any of the following errors:

- Invalid Opcode (e.g., the opcode is not listed in Figure 115);
- Invalid Parameter (e.g., the Controller ID field specifies a Controller ID not implemented in the NVM Subsystem);
- Invalid Command Size (e.g., the Request Message does not contain a complete command); or
- Invalid Command Input Data Size (e.g., the Request Data field is larger than the size specified in the Data Length field).

If the NVMe Admin Command Request Message is well formed, then a Success Response is transmitted. The Success Response contains the status associated with NVMe Admin Command in the Status Field of Completion Queue Entry Dword 3. The Status Field contains any NVM Express Base and I/O Command Set specifications specific status codes (e.g., Success or Invalid Parameter).

6.3 Get Log Page

Figure 122 defines the log pages that are mandatory, optional, and prohibited for SMBus/I2C and PCIe VDM Management Endpoint on NVMe Storage Devices and NVMe Enclosures.

Figure 122: Management Endpoint - Log Page Support

Log Page Name[3]	**Log Identifier**	**SMBus/I2C Log Page Support Requirements[1]**		**PCIe VDM Log Page Support Requirements[1]**	
		NVMe Storage Device	**NVMe Enclosure**	**NVMe Storage Device**	**NVMe Enclosure**
Supported Log Pages	00h	M[2]	M[2]	M[2]	M[2]
Error Information	01h	M	M	M	M
SMART / Health Information (Controller scope)	02h	M	O	M	O
SMART / Health Information (NVM Subsystem scope)		O	O	O	O
Firmware Slot Information	03h	M	O	M	O
Changed Namespace List	04h	O	O	O	O
Commands Supported and Effects	05h	O	O	O	O
Device Self-test	06h	O	O	O	O
Telemetry Host-Initiated	07h	O	O	O	O
Telemetry Controller-Initiated	08h	O	O	O	O
Endurance Group Information	09h	O	O	O	O
Predictable Latency Per NVM Set	0Ah	O	O	O	O
Predictable Latency Event Aggregate	0Bh	O	O	O	O
Asymmetric Namespace Access	0Ch	O	O	O	O
Persistent Event	0Dh	O	O	O	O
LBA Status Information[4]	0Eh	O	O	O	O
Endurance Group Event Aggregate	0Fh	O	O	O	O
Media Unit Status	10h	O	O	O	O
Supported Capacity Configuration List	11h	O	O	O	O
Feature Identifiers Supported and Effects	12h	M[2]	O	M[2]	O
NVMe-MI Commands Supported and Effects	13h	O	O	O	O
Command and Feature Lockdown	14h	O	O	O	O
Boot Partition	15h	O	O	O	O
Rotational Media Information	16h	O	O	O	O
Discovery	70h	O	O	O	O
Reservation Notification	80h	O	O	O	O
Sanitize Status	81h	O	O	O	O
Changed Zone List[5]	BFh	O	O	O	O

Notes:

1. O = Optional, M = Mandatory, P = Prohibited.
2. Optional for versions 1.1 and earlier of this specification.
3. Refer to the NVM Express Base Specification unless another footnote specifies otherwise.
4. Refer to the NVM Command Set Specification.
5. Refer to the Zoned Namespace Command Set Specification.

6.4 Sanitize Operation

Figure 123 specifies the Command Messages allowed during a sanitize operation and the Command Messages that should be allowed during the processing of a Format NVM command. Refer to the NVM Express Base Specification for the definition of a sanitize operation.

Figure 123: Command Messages Allowed During Sanitize Operation and During the Processing of a Format NVM Command

Command Set	Command Message	Allowed
Management Interface Command Set	Configuration Get	Yes
	Configuration Set	
	Controller Health Status Poll	
	Management Endpoint Buffer Read	
	Management Endpoint Buffer Write	
	NVM Subsystem Health Status Poll	
	Read NVMe-MI Data Structure	
	Reset	
	SES Receive	
	SES Send	
	Shutdown	
	VPD Read	
	VPD Write	
NVMe Admin Command Set [1]	Capacity Management	Same restrictions as defined by the NVM Express Base Specification
	Device Self-test	
	Firmware Activate/Commit	
	Firmware Image Download	
	Format NVM	
	Get Features	
	Get Log Page	
	Identify	
	Namespace Attachment	
	Namespace Management	
	Sanitize	
	Security Receive/Send	
	Security Send	
	Set Features	
	Vendor Specific	
	Virtualization Management	
PCIe Command Set	PCIe Configuration Read	Yes
	PCIe Configuration Write	
	PCIe I/O Read	
	PCIe Memory Read	
	PCIe Memory Write	

NOTES:

1. NVMe Admin Commands that are prohibited via the out-of-band mechanism (refer to Figure 115) are not listed since they are always prohibited including during a sanitize operation.

6.5 Set Features and Get Features

Figure 124 defines features that are mandatory or optional for an I/O Controller. Refer to the NVM Express Base Specification for the definition of Set Features and Get Features commands and I/O Controllers.

All Feature Identifiers supported shall be supported if received in-band on an NVMe Controller or received out-of-band on a Management Endpoint unless otherwise stated.

Figure 124: I/O Controller – Feature Support

Feature Name	Feature Support Requirements[1]	Logged in Persistent Event Log[1]
Enhanced Controller Metadata	M	O
Controller Metadata	M	O
Namespace Metadata	M	O
Notes: 1. O = Optional, M = Mandatory		

Figure 125 defines features that are mandatory or optional for an Administrative Controller. Refer to the NVM Express Base Specification for the definition of Set Features and Get Features commands and Administrative Controller.

Figure 125: Administrative Controller – Feature Support

Feature Name	Feature Support Requirements[1]	Logged in Persistent Event Log[1]
Enhanced Controller Metadata	M	O
Controller Metadata	M	O
Namespace Metadata	O	O
Notes: 1. O = Optional, M = Mandatory		

Figure 126 define the features that are mandatory, optional, and prohibited for SMBus/I2C and PCIe VDM Management Endpoints on NVMe Storage Devices and NVMe Enclosures.

Figure 126: Management Endpoint - Feature Support

Feature Name[2]	Feature Identifier	SMBus/I2C Feature Support Requirements[1]		PCIe VDM Feature Support Requirements[1]	
		NVMe Storage Device	NVMe Enclosure	NVMe Storage Device	NVMe Enclosure
Arbitration	01h	P	P	P	P
Power Management	02h	O	O	O	O
LBA Range Type[3]	03h	P	P	P	P
Temperature Threshold	04h	O	O	O	O
Error Recovery[3]	05h	P	P	P	P
Volatile Write Cache	06h	P	P	P	P
Number of Queues	07h	P	P	P	P
Interrupt Coalescing	08h	P	P	P	P
Interrupt Vector Configuration	09H	P	P	P	P
Write Atomicity Normal[3]	0Ah	P	P	P	P
Asynchronous Event Configuration	0Bh	P	P	P	P
Autonomous Power State Transition	0Ch	O	O	O	O
Host Memory Buffer	0Dh	P	P	P	P
Timestamp	0Eh	O	O	O	O
Keep Alive Timer	0Fh	P	P	P	P

Figure 126: Management Endpoint - Feature Support

Feature Name[2]	Feature Identifier	SMBus/I2C Feature Support Requirements[1]		PCIe VDM Feature Support Requirements[1]	
		NVMe Storage Device	NVMe Enclosure	NVMe Storage Device	NVMe Enclosure
Host Controlled Thermal Management	10h	O	O	O	O
Non-Operational Power State Config	11h	O	O	O	O
Read Recovery Level Config	12h	P	P	P	P
Predictable Latency Mode Config	13h	P	P	P	P
Predictable Latency Mode Window	14h	P	P	P	P
LBA Status Information Attributes[3]	15h	P	P	P	P
Host Behavior Support	16h	P	P	P	P
Sanitize Config	17h	O	O	O	O
Endurance Group Event Configuration	18h	P	P	P	P
I/O Command Set Profile	19h	O	P	O	P
Spinup Control	1Ah	O	O	O	O
Key Value Configuration[4]	20h	O	O	O	O
Enhanced Controller Metadata	7Dh	O	O	O	O
Controller Metadata	7Eh	O	O	O	O
Namespace Metadata	7Fh	O	O	O	O
Software Progress Marker	80h	P	P	P	P
Host Identifier	81h	P	P	P	P
Reservation Notification Mask	82h	P	P	P	P
Reservation Persistence	83h	P	P	P	P
Namespace Write Protection Config	84h	P	P	P	P

Notes:

1. O = Optional, M = Mandatory, P = Prohibited for Set Features/Optional for Get Features.
2. Refer to the NVM Express Base Specification unless another footnote specifies otherwise.
3. Refer to the NVM Command Set Specification.
4. Refer to the Key Value Command Set Specification.

7 PCIe Command Set (Optional)

The PCIe Command Set defines commands that a Management Controller may submit to access the memory, I/O, and configuration addresses spaces associated with a Controller in the NVM Subsystem. Only addresses mapped to the specified Controller may be accessed (e.g., these commands do not directly access memory on a host). The NMIMT field in the message header for PCIe Command Messages and Response Messages is set to 4h (PCIe Command). The PCIe Command Set is only applicable in the out-of-band mechanism and is prohibited in the in-band tunneling mechanism. The processing of commands in the PCIe Command Set may be affected by the Command and Feature Lockdown feature (refer to the NVM Express Base Specification).

PCIe Commands over the out-of-band mechanism may interfere with host software. A Management Controller should coordinate with the host or issue only PCIe Commands that do not interfere with host software or in-band NVMe commands (e.g., PCIe Configuration Read). Coordination between a Management Controller and a host is outside the scope of this specification.

The Request Message format for PCIe Commands is shown in Figure 127 and described in Figure 128.

Figure 127: PCIe Command Request Format

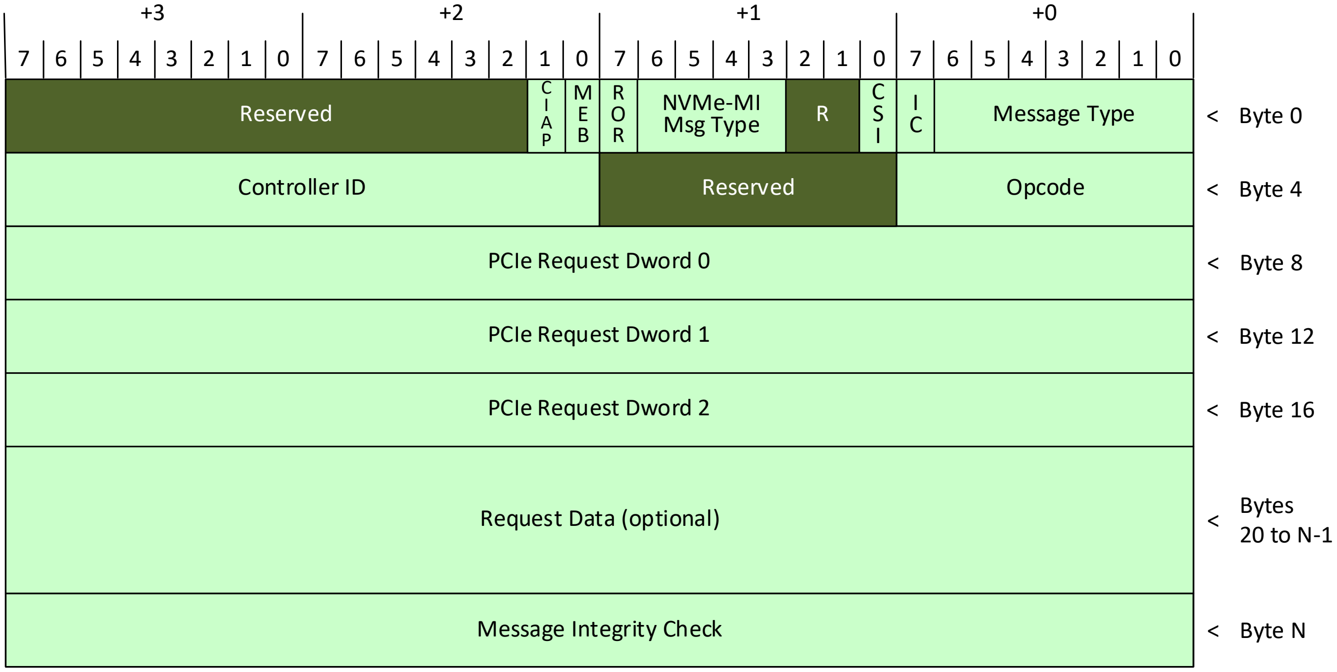


Figure 128: PCIe Command Request Description

Bytes	Description
03:00	**NVMe-MI Message Header (NMH):** Refer to section 3.1.
04	**Opcode (OPC):** This field specifies the opcode of the command to be processed. Refer to Figure 129.
05	Reserved
07:06	**Controller ID (CTLID):** This field specifies the Controller ID of the NVMe Controller that this command targets.
11:08	**PCIe Request Dword 0 (NMD0):** This field is command specific Dword 0.
15:12	**PCIe Request Dword 1 (NMD1):** This field is command specific Dword 1.

Figure 128: PCIe Command Request Description

Bytes	Description
19:16	**PCIe Request Dword 2 (NMD2):** This field is command specific Dword 2.
N-1:20	**Request Data (REQD):** (Optional)
N+3:N	**Message Integrity Check (MIC):** Refer to section 3.1.

Figure 129 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 129: 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
05h	O	O	PCIe I/O Write
06h to FFh	-	-	Reserved

NOTES:
1. O/M/P 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 or an NVMe Enclosure and may implement optional commands allowed by either an NVMe Storage Device or an NVMe Enclosure.

The Response Message for PCIe Command is shown in Figure 130 and described in Figure 131.

Figure 130: PCIe Command Response Format

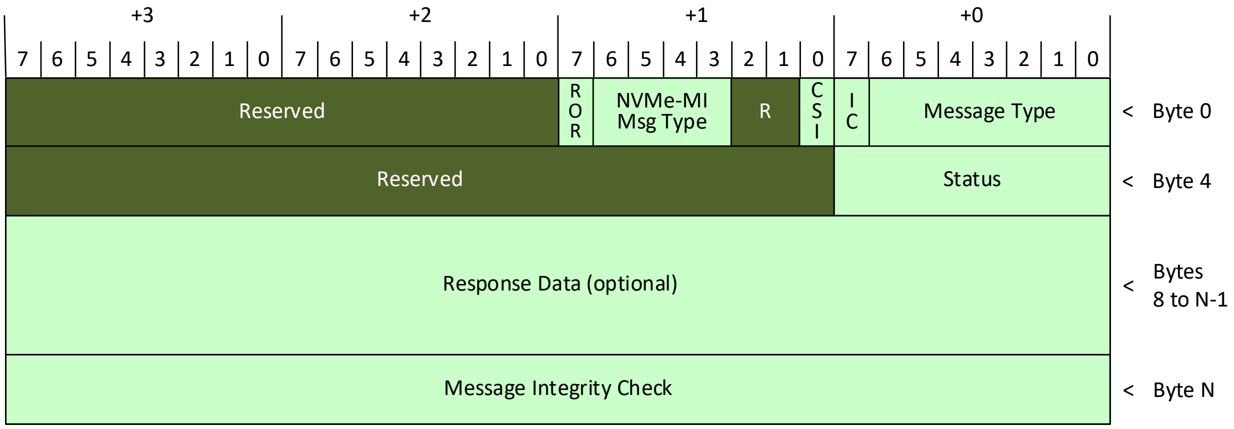


Figure 131: PCIe Command Response Description

Bytes	Description
03:00	**NVMe-MI Message Header (NMH):** Refer to section 3.1.
04	**Status (STATUS):** This field indicates the status of the PCIe Command. Refer to section 4.1.2.
07:05	Reserved
N-1:08	**Response Data (RESPD):** (Optional)
N+3:N	**Message Integrity Check (MIC):** Refer to section 3.1.

PCIe Commands allow the Management Controller to access PCI Express configuration, I/O, and memory spaces of any Controller in the NVM Subsystem. Support for PCIe Commands is optional and indicated by the Optionally Supported Commands data structure. Refer to Figure 99.

An implementation may support a subset of the PCIe Commands. For supported commands, an implementation may block access to certain address space ranges (e.g., due to security concerns). A PCIe Command that attempts to access such a blocked address range is aborted with the Status field set to Access Denied.

It is recommended that PCIe Commands provide access to all non-blocked address spaces whenever MCTP access is supported. In some implementations, it may not be possible to access PCIe resources in certain states. A PCIe Command processed when a Controller is in one of these states may be aborted with the Status field set to PCIe Inaccessible. Refer to section 8.1.

A PCIe Command that is not well-formed results in an Error Response. A PCIe Command is well formed if it does not contain any of the following errors:

- Invalid Opcode (e.g., the Opcode is not listed in Figure 129);
- Invalid Parameter (e.g., the Controller ID field specifies a Controller ID not implemented in the NVM Subsystem);
- Invalid Command Size (e.g., the Request Message does not contain a complete command); or
- Invalid Command Input Data Size (e.g., the NVMe Request Data field is larger than the size expected by the command).

7.1 PCIe Configuration Read

The PCIe Configuration Read command allows the Management Controller to read the contents of the PCIe configuration address space associated with an NVMe Controller in the NVM Subsystem. The Controller ID field in the Command Message specifies the Controller ID that is being accessed.

The command uses PCIe Request Dwords 0 and 1. PCIe Request Dword 2 is not used and is reserved. The format of PCIe Request Dwords 0 and 1 are shown in Figure 132 and Figure 133 respectively.

Figure 132: PCIe Configuration Read – PCIe Request Dword 0

Bits	Description
31:16	Reserved
15:00	**Length (LENGTH):** This field specifies the number of bytes to be read.

Figure 133: PCIe Configuration Read – PCIe Request Dword 1

Bits	Description
31:12	Reserved

Figure 133: PCIe Configuration Read – PCIe Request Dword 1

Bits	Description
11:00	**Offset (OFFSET):** This field specifies the offset in bytes into the 4 KiB configuration space associated with the NVMe Controller at which the read begins.

When this command is completed successfully, PCI configuration space associated with the NVMe Controller specified by Controller ID is read and returned in the Response Data field. The Offset field specifies the starting read offset in PCIe configuration address space and the Length field specifies the number of bytes to be read. The Response Data field is always an integral number of dwords and is equal to the Length field rounded up to the next dword. If Length is not an integral number of dwords, then the read data shall be padded to the next dword boundary with bytes cleared to 0h.

If the sum of the Offset and Length fields fall outside of PCI configuration space, then the Management Endpoint responds with an Invalid Parameter Error Response with the PEL field indicating the Offset field.

A Management Endpoint shall support the PCIe Configuration Read command if any of the other PCIe Command Set commands are supported. Access to the BAR offsets shall not return an Access Denied Response Message Status (i.e., the correct data shall be provided).

7.2 PCIe Configuration Write

The PCIe Configuration Write command allows the Management Controller to write the contents of the PCIe configuration address space associated with an NVMe Controller in the NVM Subsystem. The Controller ID field in the Command Message specifies the Controller ID that is being accessed.

The command uses PCIe Request Dwords 0 and 1. PCIe Request Dword 2 is not used and is reserved. The format of PCIe Request Dwords 0 and 1 are shown in Figure 134 and Figure 135 respectively.

Figure 134: PCIe Configuration Write – PCIe Request Dword 0

Bits	Description
31:16	Reserved
15:00	**Length (LENGTH):** This field specifies the number of bytes to be written.

Figure 135: PCIe Configuration Write – PCIe Request Dword 1

Bits	Description
31:12	Reserved
11:00	**Offset (OFFSET):** This field specifies the offset in bytes into the 4,096B configuration space associated with the NVMe Controller at which the write begins.

When this command is completed successfully, PCI configuration space associated with the NVMe Controller specified by Controller ID is written with the data contained in the Request Data field. The Offset field specifies the starting write offset in PCIe configuration address space and the Length field specifies the number of bytes to be written. The Request Data field is always an integral number of dwords and is equal to the Length field rounded up to the next dword. If Length is not an integral number of dwords, then unused padding bytes are discarded.

If the sum of the Offset and Length fields fall outside of PCI configuration space, then the Management Endpoint responds with an Invalid Parameter Error Response with the PEL field indicating the Offset field.

7.3 PCIe I/O Read

The PCIe I/O Read command allows the Management Controller to read the contents of PCIe I/O space associated with an NVMe Controller in the NVM Subsystem. The Controller ID field in the Command Message specifies the Controller ID that is being accessed.

The command uses PCIe Request Dwords 0 and 1. PCIe Request Dword 2 is not used and is reserved. The format of PCIe Request Dword 0 and 1 are shown in Figure 136 and Figure 137 respectively.

Figure 136: PCIe I/O Read – PCIe Request Dword 0

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:19</td><td>Reserved</td></tr>
<tr><td>18:16</td><td><b>Base Address Register (BAR):</b> This field specifies the PCI Base Address Register (BAR) of the I/O space to be read. BARs are located beginning at 10h in PCI Configuration space (refer to the NVMe over PCIe Transport Specification) and the value of this field specifies the starting offset of the associated BAR. For a 64-bit BAR, this field should correspond to the least-significant 32-bits of the BAR.
<table>
<tr><th>Value</th><th>BAR Offset</th></tr>
<tr><td>0h</td><td>10h</td></tr>
<tr><td>1h</td><td>14h</td></tr>
<tr><td>2h</td><td>18h</td></tr>
<tr><td>3h</td><td>1Ch</td></tr>
<tr><td>4h</td><td>20h</td></tr>
<tr><td>5h</td><td>24h</td></tr>
<tr><td>6h to 7h</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>15:00</td><td><b>Length (LENGTH):</b> This field specifies the number of bytes to be read.</td></tr>
</table>

Figure 137: PCIe I/O Read – PCIe Request Dword 1

Bits	Description
31:00	**Offset (OFFSET):** This field specifies the offset in bytes into the PCI BAR associated with the NVMe Controller at which the read begins.

When this command is completed successfully, PCI I/O space associated with the NVMe Controller specified by Controller ID is read and returned in the Response Data field. The Offset field specifies the starting read offset in PCIe I/O address space specified by the Base Address Register field. The Length field specifies the number of bytes to be read. The Response Data field is always an integral number of dwords and is equal to the Length field rounded up to the next dword. If Length is not an integral number of dwords, then the read data shall be padded to the next dword boundary with bytes cleared to 0h.

If the Base Address Register field does not correspond to an I/O BAR implemented by the specified NVMe Controller, then the Management Endpoint responds with an Invalid Parameter Error Response with the PEL field indicating the Base Address Register field.

If the sum of the Offset and Length fields fall outside the address range of the BAR specified by the Base Address Register field, then the Management Endpoint responds with an Invalid Parameter Error Response with the PEL field indicating the Offset field.

7.4 PCIe I/O Write

The PCIe I/O Write command allows the Management Controller to write the contents of PCIe I/O space associated with an NVMe Controller in the NVM Subsystem. The Controller ID field in the Command Message specifies the Controller ID that is being accessed.

The command uses PCIe Request Dwords 0 and 1. PCIe Request Dword 2 is not used and is reserved. The format of PCIe Request Dword 0 and 1 are shown in Figure 138 and Figure 139 respectively.

Figure 138: PCIe I/O Write – PCIe Request Dword 0

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:19</td><td>Reserved</td></tr>
<tr><td>18:16</td><td><b>Base Address Register (BAR):</b> This field specifies the PCI Base Address Register (BAR) of the I/O space to be written. BARs are located beginning at 10h in PCI Configuration space (refer to the NVMe over PCIe Transport Specification) and the value of this field specifies the starting offset of the associated BAR. For a 64-bit BAR, this field should correspond to the least-significant 32-bits of the BAR.
<table>
<tr><th>Value</th><th>BAR Offset</th></tr>
<tr><td>0h</td><td>10h</td></tr>
<tr><td>1h</td><td>14h</td></tr>
<tr><td>2h</td><td>18h</td></tr>
<tr><td>3h</td><td>1Ch</td></tr>
<tr><td>4h</td><td>20h</td></tr>
<tr><td>5h</td><td>24h</td></tr>
<tr><td>6h to 7h</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>15:00</td><td><b>Length (LENGTH):</b> This field specifies the number of bytes to be written.</td></tr>
</table>

Figure 139: PCIe I/O Write – PCIe Request Dword 1

Bits	Description
31:00	**Offset (OFFSET):** This field specifies the offset in bytes into the PCI BAR associated with the NVMe Controller at which the write begins.

When this command is completed successfully, PCI I/O space associated with the NVMe Controller specified by Controller ID is written with the data contained in the Request Data field. The Offset field specifies the starting write offset in PCIe I/O address space specified by the Base Address Register field. The Length field specifies the number of bytes to be written. The Request Data field is always an integral number of dwords and is equal to the Length field rounded up to the next dword. If Length is not an integral number of dwords, then unused padding bytes are discarded.

If the Base Address Register field does not correspond to an I/O BAR implemented by the specified NVMe Controller, then the Management Endpoint responds with an Invalid Parameter Error Response with the PEL field indicating the Base Address Register field.

If the sum of the Offset and Length fields fall outside the address range of the BAR specified by the Base Address Register field, then the Management Endpoint responds with an Invalid Parameter Error Response with the PEL field indicating the Offset field.

7.5 PCIe Memory Read

The PCIe Memory Read command allows the Management Controller to read the contents of PCIe memory associated with an NVMe Controller in the NVM Subsystem. The Controller ID field in the Command Message specifies the Controller ID that is being accessed.

The command uses PCIe Request Dwords 0, 1, and 2. The format of PCIe Request Dword 0, 1, and 2 are shown in Figure 140, Figure 141, and Figure 142 respectively.

Figure 140: PCIe Memory Read – PCIe Request Dword 0

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:19</td><td>Reserved</td></tr>
<tr><td>18:16</td><td><b>Base Address Register (BAR):</b> This field specifies the PCI Base Address Register (BAR) of the memory space to be read. BARs are located beginning at 10h in PCI Configuration space (refer to the NVMe over PCIe Transport Specification) and the value of this field specifies the starting offset of the associated BAR. For a 64-bit BAR, this field should correspond to the least-significant 32-bits of the BAR.
<table>
<tr><th>Value</th><th>BAR Offset</th></tr>
<tr><td>0h</td><td>10h</td></tr>
<tr><td>1h</td><td>14h</td></tr>
<tr><td>2h</td><td>18h</td></tr>
<tr><td>3h</td><td>1Ch</td></tr>
<tr><td>4h</td><td>20h</td></tr>
<tr><td>5h</td><td>24h</td></tr>
<tr><td>6h to 7h</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>15:00</td><td><b>Length (LENGTH):</b> This field specifies the number of bytes to be read.</td></tr>
</table>

Figure 141: PCIe Memory Read – PCIe Request Dword 1

Bits	Description
31:00	**Offset (OFFSET):** This field specifies the least-significant 32-bits (i.e., bit 0 to bit 31) of the offset in bytes into the PCI BAR associated with the NVMe Controller at which the read begins.

Figure 142: PCIe Memory Read – PCIe Request Dword 2

Bits	Description
31:00	**Offset (OFFSET):** This field specifies the most-significant 32-bits (i.e., bit 32 to bit 63) of the offset in bytes into the PCI BAR associated with the NVMe Controller at which the read begins.

When this command is completed successfully, PCI memory space associated with the NVMe Controller specified by Controller ID is read and returned in the Response Data field. The Offset field specifies the starting read offset in PCIe memory address space specified by the Base Address Register field. The Length field specifies the number of bytes to be read. The Response Data field is always an integral number of dwords and is equal to the Length field rounded up to the next dword. If Length is not an integr al number of dwords, then the read data shall be padded to the next dword boundary with bytes cleared to 0h.

If the Base Address Register field does not correspond to one implemented by the specified NVMe Controller, or the address range specified by the Base Address Range is not a memory region, then the Management Endpoint responds with an Invalid Parameter Error Response with the PEL field indicating the Base Address Register field.

If the sum of the Offset and Length fields fall outside the address range specified by the Base Address Register field, then the Management Endpoint responds with an Invalid Parameter Error Response with the PEL field indicating the Offset field.

7.6 PCIe Memory Write

The PCIe Memory Write command allows the Management Controller to write the contents of PCIe memory associated with an NVMe Controller in the NVM Subsystem. The Controller ID field in the Command Message specifies the Controller ID that is being accessed.

The command uses PCIe Request Dwords 0, 1, and 2. The format of PCIe Request Dword 0, 1, and 2 are shown in Figure 143, Figure 144, and Figure 145 respectively.

Figure 143: PCIe Memory Write – PCIe Request Dword 0

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:19</td><td>Reserved</td></tr>
<tr><td>18:16</td><td><b>Base Address Register (BAR):</b> This field specifies the PCI Base Address Register (BAR) of the memory space to be written. BARs are located beginning at 10h in PCI Configuration space (refer to the NVMe over PCIe Transport Specification) and the value of this field specifies the starting offset of the associated BAR. For a 64-bit BAR, this field should correspond to the least-significant 32-bits of the BAR.
<table>
<tr><th>Value</th><th>BAR Offset</th></tr>
<tr><td>0h</td><td>10h</td></tr>
<tr><td>1h</td><td>14h</td></tr>
<tr><td>2h</td><td>18h</td></tr>
<tr><td>3h</td><td>1Ch</td></tr>
<tr><td>4h</td><td>20h</td></tr>
<tr><td>5h</td><td>24h</td></tr>
<tr><td>6h to 7h</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>15:00</td><td><b>Length (LENGTH):</b> This field specifies the number of bytes to be written.</td></tr>
</table>

Figure 144: PCIe Memory Write – PCIe Request Dword 1

Bits	Description
31:00	**Offset (OFFSET):** This field specifies the least-significant 32-bits (i.e., bit 0 to bit 31) of the offset in bytes into the PCI BAR associated with the NVMe Controller at which the write begins.

Figure 145: PCIe Memory Write – PCIe Request Dword 2

Bits	Description
31:00	**Offset (OFFSET):** This field specifies the most-significant 32-bits (i.e., bit 32 to bit 63) of the offset in bytes into the PCI BAR associated with the NVMe Controller at which the write begins.

When this command is completed successfully, PCI memory space associated with the NVMe Controller specified by Controller ID is written with the data contained in the Request Data field. The Offset field specifies the starting write offset in PCIe memory address space specified by the Base Address Register field. The Length field specifies the number of bytes to be written. The Request Data field is always an integral number of dwords and is equal to the Length field rounded up to the next dword. If Length is not an integral number of dwords, then unused padding bytes are discarded.

If the Base Address Register field does not correspond to one implemented by the specified NVMe Controller, or the address range specified by the Base Address Range is not a memory region, then the Management Endpoint responds with an Invalid Parameter Error Response with the PEL field indicating the Base Address Register field.

If the sum of the Offset and Length fields fall outside the address range of the BAR specified by the Base Address Register field, then the Management Endpoint responds with an Invalid Parameter Error Response with the PEL field indicating the Offset field.

8 Management Architecture

8.1 Out-of-Band Operational Times

In the out-of-band mechanism, 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 in the absence of main power. Only the Powered On and Powered Off states are applicable in form factors and platforms that do not support auxiliary power. Figure 146 defines the power states of a Management Endpoint. Note that auxiliary power is described from the perspective of the NVM Subsystem and could be provided by any appropriate power rail in a host platform.

The operations supported in each NVM Subsystem power state are summarized in Figure 146. 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 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 146 is undefined.

Figure 146: Operations Supported During NVM Subsystem Power States

Operation	Powered Off -All Power Rails Off	Powered On -All Power Rails On	Auxiliary Power Only[2] -Main Power Off -Auxiliary Power On	Main Power Only[2] -Main Power On -Auxiliary Power Off
SMBus/I2C VPD and SMBus/I2C Mux Access	Not Supported	Supported	Supported	Implementation Specific
SMBus/I2C MCTP Access	Not Supported	Supported	Optional[1]	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.

When an NVM Subsystem transitions from a power state in which accesses are not supported to one where accesses are supported, accesses shall be processed 1 s after entering the power state in which accesses are supported. For example, an SMBus/I2C MCTP access issued 1 s after transitioning from a “Powered Off” to a “Main Power” state is guaranteed to be processed. The behavior of accesses prior to this 1 s time interval is undefined. For example, the behavior of an SMBus/I2C MCTP access issued 50 ms after transitioning from a “Powered Off” to a “Main Power” state is undefined.

When transitioning between power states in which accesses are supported in both states (i.e., the state before and after the transition), there is no interruption in access processing (i.e., accesses are processed prior to the state transition, during the state transition, and immediately after entering the new power state).

Although not recommended, an implementation may choose not to support processing of PCIe Commands that target a Controller in the NVM Subsystem that is in any of the following states:[2]

- Controller Level Reset;
- SR-IOV virtual function is not enabled;
- During any type of PCI Express Conventional Reset;
- During a PCI Express Function Level Reset (FLR);
- When the PCI Express Function is in a non-D0 power D-state; or
- When the PCI Express link is down (i.e., not in the DL_Active state).

If a PCIe Command is received that targets a Controller in one of these states and the implementation does not support processing of PCIe Commands in that state, then the PCIe command is completed with status PCIe Inaccessible. Processing of supported PCIe Commands is required in all other Controller states.

If a PCIe Command is received that targets a Controller whose corresponding PCIe link is in a low power state (i.e., PCIe ASPM), then processing of the command may cause the link to temporarily exit the low power state.

8.2 Vital Product Data

The Vital Product Data (VPD) is FRU Information (refer to the IPMI Platform Management FRU Information Storage Specification) describing an NVMe Storage Device. Each NVMe Storage Device FRU shall have a FRU Information Device with a size of 256 to 4,096 bytes which contains the VPD. The VPD for NVMe Storage Device FRUs shall contain the required elements defined in Figure 147. The VPD and FRU Information Device are optional for:

a) NVMe Storage Devices that are not FRUs (e.g., NVMe Storage Devices with a Form Factor type of Integrated per Figure 160); and
b) NVMe Enclosures.

The VPD contents for these optional use cases is outside the scope of this specification.

Figure 147: VPD Elements

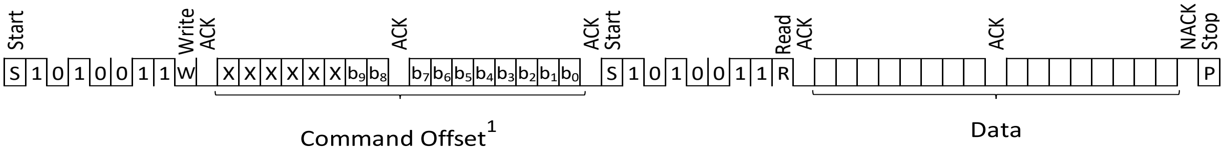
Byte	Name
7:0	Common Header
Vendor Specific	Product Info Area (Optional)
Vendor Specific	MultiRecord Info Area
Vendor Specific	Internal Use Area (Optional)
Vendor Specific	Chassis Info Area (Optional)
Vendor Specific	Board Info Area (Optional)

[2] A Management Controller shall only send these commands using SMBus/I2C or another PCIe port since the link associated with the PCIe port and Controller is down in these states.

The VPD shall be accessible using the VPD Read command on all Management Endpoints on the NVMe Storage Device FRU. The entire contents of the VPD may be updated using the VPD Write c ommand.

If the NVM Subsystem has an SMBus/I2C interface, then the VPD shall be accessible at the SMBus/I2C address of the FRU Information Device using I2C Reads. Updating the VPD using I2C Writes shall not be supported if the VPD Write command is supported. Refer to the IPMI Platform Management FRU Information Storage Definition for more information about the FRU Information Device access mechanisms (I2C Reads/I2C Writes) over SMBus/I2C.

Figure 148: I2C Read from a FRU Information Device



The number of valid bits in the Command Offset is dependent on the Maximum FRU Information Size. Command Offset bits that contain an "X" in Figure 148 are not valid and shall be ignored. This example shows a FRU Information Device with 10 valid Command Offset bits which corresponds to a Maximum FRU Information Device Size of 1 KiB.

Figure 148 shows an I2C Read where the A6h addresses and Command Offset are provided by the Management Controller followed by data being returned from the Management Endpoint. The Command Offset as shown in Figure 148 is stored internal to the NVMe Storage Device (i.e., the internal offset).

If an I2C Read is issued, then data is returned from the internal offset within the FRU Information Device and then the internal offset is incremented by 1h. If the Management Controller reads the last byte of the FRU Information Device (refer to Maximum FRU Information Size) via an I2C Read, then the internal offset shall be cleared to 0h (i.e., rolls over to 0h). If only one byte of the Command Offset is provided by the Management Controller, then the least-significant byte of the internal offset shall be set to that value and the most-significant byte of the internal offset shall be cleared to 0h.

The internal offset shall be cleared to 0h following a power cycle of the FRU Information Device. Implementations are allowed to maintain the current internal offset value or clear it to 0h following a reset of the FRU Information Device.

8.2.1 Common Header

The fields that make up the VPD Common Header are shown in Figure 149.

Figure 149: Common Header

Bytes	Factory Default	Description
0	01h	**IPMI Format Version Number (IPMIVER):** This field indicates the IPMI Format Version.
1	Impl Spec	**Internal Use Area Starting Offset (IUAOFF):** This field indicates the starting offset in multiples of 8 bytes for the Internal Use Area. A value of 0h may be used to indicate the Internal Use Area is not present.
2	Impl Spec	**Chassis Info Area Starting Offset (CIAOFF):** This field indicates the starting offset in multiples of 8 bytes for the Chassis Info Area. A value of 0h may be used to indicate the Chassis Info Area is not present.

Figure 149: Common Header

Bytes	Factory Default	Description
3	Impl Spec	**Board Info Area Starting Offset (BIAOFF):** This field indicates the starting offset in multiples of 8 bytes for the Board Info Area. A value of 0h may be used to indicate the Board Info Area is not present.
4	Impl Spec	**Product Info Area Starting Offset (PIAOFF):** This field indicates the starting offset in multiples of 8 bytes for the Product Info Area. A value of 0h may be used to indicate the Product Info Area is not present.
5	Impl Spec	**MultiRecord Info Area Starting Offset (MRIOFF):** This field indicates the starting offset in multiples of 8 bytes for the MultiRecord Info Area.
6	00h	Reserved
7	Impl Spec	**Common Header Checksum (CHCHK):** Checksum computed over byte 0 to byte 6. The checksum is computed by adding the 8-bit value of the bytes 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.

8.2.2 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 1.7. The Product Info Area factory default values shall be set to the values defined in Figure 151. The Type/Length bytes use the format shown in Figure 150.

Figure 150: 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 0h indicates that the field is empty

Figure 151: 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.
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.
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.
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.

Figure 151: Product Info Area Factory Default Values

Factory Default	Description
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.
Impl Spec	**Product Version Type/Length (PVTL):** This field indicates the type and length of the Product Version 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.
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 0h 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 0h may be used to indicate a 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.

8.2.3 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. Implementations compliant to version 1.1 and later of this specification should implement the Topology MultiRecord (refer to section 8.2.5). For backwards compatibility, the NVMe MultiRecord and the NVMe PCIe Port MultiRecord (refer to section 8.2.4) should both be included in the VPD in addition to the Topology MultiRecord unless the NVMe Storage Device FRU has Expansion Connectors, has more than one NVM Subsystem, or if including both this MultiRecord and the NVMe PCIe Port MultiRecord would extend the size of the VPD beyond 256 bytes. If either the NVMe MultiRecord or NVMe PCIe Port MultiRecord are not included, then neither MultiRecord should be included.

Figure 152: NVMe MultiRecord Area

<table>
<tr><th>Bytes</th><th>Factory Default</th><th>Description</th></tr>
<tr><td>00</td><td>0Bh</td><td>NVMe Record Type ID</td></tr>
<tr><td>01</td><td>02h or 82h</td><td><b>Record Format:</b>
<table>
<tr><th>Bits</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>
</table>

Figure 152: NVMe MultiRecord Area

Bytes	Factory Default	Description
02	20h or 3Bh	**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.
03	Impl Spec	**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 0h).
04	Impl Spec	**Header Checksum:** This field is used to give the record header a zero checksum (i.e., the modulo 256 sum of the least-significant byte of the header through this checksum byte equals 0h).
05	0h	**NVMe MultiRecord Area Version Number:** This field indicates the version number of this NVMe MultiRecord. This field shall be cleared to 0h in this version of the specification.
06	Impl Spec	**Form Factor (FF):** This field indicates the form factor of the Management Endpoint. Refer to the values in Figure 160.
12:07	0h	Reserved
13	Impl Spec [1]	**Initial 1.8 V Power Supply Requirements:** This field specifies the initial 1.8 V power supply requirements in Watts prior to receiving a Set Slot Power message.
14	Impl Spec [1]	**Maximum 1.8 V Power Supply Requirements:** This field specifies the maximum 1.8 V power supply requirements in Watts.
15	Impl Spec [1]	**Initial 3.3 V Power Supply Requirements:** This field specifies the initial 3.3 V power supply requirements in Watts prior to receiving a Set Slot Power message.
16	Impl Spec [1]	**Maximum 3.3 V Power Supply Requirements:** This field specifies the maximum 3.3 V power supply requirements in Watts.
17	0h	Reserved
18	Impl Spec [1]	**Maximum 3.3 V aux Power Supply Requirements:** This field specifies the maximum 3.3 V power supply requirements in 10 mW units.
19	Impl Spec [1]	**Initial 5 V Power Supply Requirements:** This field specifies the initial 5 V power supply requirements in Watts prior to receiving a Set Slot Power message.
20	Impl Spec [1]	**Maximum 5 V Power Supply Requirements:** This field specifies the maximum 5 V power supply requirements in Watts.
21	Impl Spec [1]	**Initial 12 V Power Supply Requirements:** This field specifies the initial 12 V power supply requirements in Watts prior to receiving a Set Slot Power message.
22	Impl Spec [1]	**Maximum 12 V Power Supply Requirements:** This field specifies the maximum 12 V power supply requirements in Watts.
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	0h	If the RLEN field is set to 3Bh, then this field is reserved. If the RLEN field is set to 20h, then this field is not present.

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.

8.2.4 NVMe PCIe Port MultiRecord Area

This MultiRecord is used to describe the PCIe connectivity for NVMe Storage Devices with a single NVM Subsystem. Implementations compliant to version 1.1 and later of this specification should implement the Topology MultiRecord (refer to section 9.2.5). For backwards compatibility, the NVMe PCIe Port MultiRecord and the NVMe MultiRecord (refer to section 8.2.3) should both be included in the VPD in addition to the Topology MultiRecord unless the NVMe Storage Device FRU has Expansion Connectors, has more than one NVM Subsystem, or if including both this MultiRecord and the NVMe MultiRecord would extend the size of the VPD beyond 256 bytes. If either the NVMe MultiRecord or NVMe PCIe Port MultiRecord are not included then neither MultiRecord should be included.

Figure 153: NVMe PCIe Port MultiRecord Area

<table>
<tr><th>Bytes</th><th>Factory Default</th><th>Description</th></tr>
<tr><td>00</td><td>0Ch</td><td><b>NVMe PCIe Port Record Type ID</b></td></tr>
<tr><td>01</td><td>02h or 82h</td><td><b>Record Format:</b>
<table>
<tr><th>Bits</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>08h or 0Bh</td><td><b>Record Length (RLEN):</b> 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><b>Record Checksum:</b> 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 0h).</td></tr>
<tr><td>04</td><td>Impl Spec</td><td><b>Header Checksum:</b> This field is used to give the record header a zero checksum (i.e., the modulo 256 sum of the least-significant byte of the header through this checksum byte equals 0h).</td></tr>
<tr><td>05</td><td>1h</td><td><b>NVMe PCIe Port MultiRecord Area Version Number:</b> This field indicates the version number of this NVMe PCIe Port MultiRecord. This field shall be set to 1h in this version of the specification.</td></tr>
<tr><td>06</td><td>Impl Spec</td><td><b>PCIe Port Number:</b> 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><b>Port Information:</b> This field indicates information about the PCIe Ports in the device.
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>7:1</td><td>Reserved</td></tr>
<tr><td>0</td><td>If this bit is set to ‘1’, then all PCIe ports within the device have the same capabilities (i.e., the capabilities listed in this structure are consistent across each PCIe port). If this bit is cleared to ‘0’, then all PCIe ports within the device do not have the same capabilities.</td></tr>
</table></td></tr>
<tr><td>08</td><td>Impl Spec</td><td><b>PCIe Link Speed:</b> This field indicates a bit vector of link speeds supported by the PCIe port.
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>7:5</td><td>Reserved</td></tr>
<tr><td>4</td><td>Set to ‘1’ if the PCIe link supports 32.0 GT/s, otherwise cleared to ‘0’.</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>
</table>

Figure 153: NVMe PCIe Port MultiRecord Area

<table>
<tr><th>Bytes</th><th>Factory Default</th><th>Description</th></tr>
<tr><td>09</td><td>Impl Spec</td><td><b>PCIe Maximum Link Width:</b> 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 Requester 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 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><b>MCTP Support:</b> This field contains a bit vector that specifies the level of support for the NVMe Management Interface.
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>7:1</td><td>Reserved</td></tr>
<tr><td>0</td><td>If this bit is set to ‘1’, then MCTP-based management commands are supported on the PCIe port. If this bit is cleared to ‘0’, then MCTP-based management commands are not supported on the PCIe port.</td></tr>
</table></td></tr>
<tr><td>11</td><td>Impl Spec</td><td><b>Ref Clk Capability:</b> This field contains a bit vector that specifies the PCIe clocking modes supported by the port.
<table>
<tr><th>Bits</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 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>12</td><td>Impl Spec</td><td><b>Port Identifier:</b> This field contains the NVMe-MI Port Identifier.</td></tr>
<tr><td>15:13</td><td>0h</td><td>If the RLEN field is set to 0Bh, then this field is reserved. If the RLEN field is set to 08h, then this field is not present.</td></tr>
</table>

8.2.5 Topology MultiRecord Area

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

The Topology MultiRecord consists mainly of a list of Element Descriptors as shown in Figure 154. 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 156 and Element Descriptor Types are listed in Figure 157.

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 154: Topology MultiRecord

<table>
<tr><th>Bytes</th><th>Factory Default</th><th>Description</th></tr>
<tr><td>00</td><td>0Dh</td><td><b>Topology Record Type ID</b></td></tr>
<tr><td>01</td><td>2h or 82h</td><td><b>Record Format:</b>
<table>
<tr><th>Bits</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 shall be set to 2h.</td></tr>
</table></td></tr>
<tr><td>02</td><td>Impl Spec</td><td><b>Record Length (RLEN):</b> 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><b>Record Checksum:</b> 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 0h).</td></tr>
<tr><td>04</td><td>Impl Spec</td><td><b>Header Checksum:</b> This field is used to give the record header a zero checksum (i.e., the modulo 256 sum of the least-significant byte of the header through this checksum byte equals 0h).</td></tr>
<tr><td>05</td><td>0h</td><td><b>Version Number:</b> This field indicates the version number of this Topology MultiRecord. This field shall be cleared to 0h in this version of the specification.</td></tr>
<tr><td>06</td><td>0h</td><td>Reserved</td></tr>
<tr><td>07</td><td>Impl Spec</td><td><b>Element Count (N):</b> 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><b>Element Descriptor 0:</b> This field contains the first Element Descriptor in this Topology MultiRecord.</td></tr>
<tr><td>Impl Spec</td><td>Impl Spec</td><td><b>Element Descriptor 1:</b> 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 155 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 155: Indexing Across Extended MultiRecords

Index	Topology Multi Record 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 8.2.5.1.

Figure 156: Element Descriptor

Bytes	Factory Default	Description
00	Impl Spec	**Type:** This field indicates the type of the Element Descriptor. Values are defined in Figure 157.
01	Impl Spec	**Revision:** This field indicates the revision of the Element Descriptor.
02	Impl Spec	**Length:** Number of bytes in the Element Descriptor.
Length - 1:03	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 157. Subsequent sections define the details associated with each Element Descriptor Type.

Figure 157: Element Descriptor Types

Value	Name	Reference Section
0	Reserved	-
1	**Extended Element Descriptor**	8.2.5.1
2	**Upstream Connector Element Descriptor**	8.2.5.2
3	**Expansion Connector Element Descriptor**	8.2.5.3
4	**Label Element Descriptor**	8.2.5.4
5	**SMBus/I2C Mux Element Descriptor**	8.2.5.5
6	**PCIe Switch Element Descriptor**	8.2.5.6
7	**NVM Subsystem Element Descriptor**	8.2.5.7
8	**FRU Information Device Element Descriptor**	8.2.5.8
9 to 239	Reserved	-
240 to 255	**Vendor specific**	8.2.5.9

8.2.5.1 Extended Element Descriptor

The Extended Element Descriptor is shown in Figure 158. 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 155 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 158: Extended Element Descriptor

Bytes	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.

8.2.5.2 Upstream Connector Element Descriptor

The Upstream Connector Element Descriptor is shown in Figure 159 and is used to describe an Upstream Connector (i.e., a connector through which a Requester communicates with the NVMe Storage Device). Upstream Element Descriptors are always a parent and never a child.

Figure 159: Upstream Connector Element Descriptor

Bytes	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 160 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.

Figure 159: Upstream Connector Element Descriptor

Bytes	Factory Default	Description
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 field 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 160 lists the NVMe Storage Device's Form Factor values.

Figure 160: Form Factors

Value	Description	
	Interface	Form Factor Description
0	Unspecified	Other – unknown
1	PCIe	Integrated
2	PCIe	Other - unknown
3 to 15	Reserved	
16	PCIe	2.5" Form Factor – unknown
17	PCIe	2.5" Form Factor – PCI Express SFF-8639 Module (U.2) 15 mm
18	PCIe	2.5" Form Factor – PCI Express SFF-8639 Module (U.2) 7 mm
19	PCIe	2.5" Form Factor – (SFF-TA-1001) 15 mm
20	PCIe	2.5" Form Factor – (SFF-TA-1001) 7 mm
21 to 31	Reserved	
32	PCIe	CEM add in card – unknown
33	PCIe	CEM add in card – Low Profile (HHHL)
34	PCIe	CEM add in card – Standard Height Half Length (FHHL)
35	PCIe	CEM add in card – Standard Height Full Length (FHFL)
36 to 47	Reserved	
48	PCIe	M.2 module – unknown
49	PCIe	M.2 module – 2230
50	PCIe	M.2 module – 2242
51	PCIe	M.2 module – 2260
52	PCIe	M.2 module – 2280
53	PCIe	M.2 module – 22110
54 to 63	Reserved	
64	PCIe	BGA SSD – unknown
65	PCIe	BGA SSD – 16 x 20mm (M.2 Type 1620)
66	PCIe	BGA SSD – 11.5 x 13mm (M.2 Type 1113)
67 to 79	Reserved	
80	PCIe	Enterprise & Datacenter SSD Form Factor – unknown
81	PCIe	E1.S - (SFF-TA-1006) 5.9 mm
82	PCIe	E1.S - (SFF-TA-1006) 8 mm
83	PCIe	E1.L - (SFF-TA-1007) 9.5 mm
84	PCIe	E1.L - (SFF-TA-1007) 18 mm
85	PCIe	E3.S - (SFF-TA-1008) 7.5 mm

Figure 160: Form Factors

Value	Description	
	Interface	Form Factor Description
86	PCIe	E3.S - (SFF-TA-1008) 16.8 mm
87	PCIe	E3.L - (SFF-TA-1008) 7.5 mm
88	PCIe	E3.L - (SFF-TA-1008) 16.8 mm
89	PCIe	E1.S - (SFF-TA-1006) 9.5 mm
90	PCIe	E1.S - (SFF-TA-1006) 15 mm
91	PCIe	E1.S - (SFF-TA-1006) 25 mm
92 to 95	Reserved	
96	Ethernet	Other – unknown
97	Ethernet	2.5" Form Factor – (Native NVMe-oF Drive) 15 mm
98	Ethernet	2.5" Form Factor – (Native NVMe-oF Drive) 7 mm
99	Ethernet	E3.S – (Native NVMe-oF Drive) 7.5 mm
100	Ethernet	E3.S – (Native NVMe-oF Drive) 16.8 mm
101 to 239	Reserved	
240 to 255	Vendor Specific	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 Requester 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 161. It contains a list of pointers to child Element Descriptors whose SMBus/I2C port is directly connected to the Upstream Connector.

Figure 161: SMBus/I2C Upstream Port Descriptor

Bytes	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 the Count field 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 the Count field is not greater than one, then this field is not present.
…	…	…

A PCIe Upstream Port Descriptor is shown in Figure 162. 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.

Figure 162: PCIe Upstream Port Descriptor

Bytes	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.
…	…	…

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); and
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; and
4. PCIe lanes 12 to 15.

8.2.5.3 Expansion Connector Element Descriptor

The Expansion Connector Element Descriptor is shown in Figure 163 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 163: Expansion Connector Element Descriptor

Bytes	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 FRU that plugs into the Expansion Connector. Refer to Figure 160 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.
Impl Spec	Impl Spec	**Expansion Connector Port Descriptor 0:** This field contains the first Expansion Connector Port Descriptor.
Impl Spec	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 164. 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.

Figure 164: Expansion Connector PCIe Port Descriptor

Bytes	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.

Figure 164: Expansion Connector PCIe Port Descriptor

Bytes	Factory Default	Description
03	Impl Spec	**Ending Lane:** This field indicates the ending PCIe lane of the port on the Expansion Connector PCIe Port Descriptor.

8.2.5.4 Label Element Descriptor

The Label Element Descriptor is shown in Figure 165 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 165: Label Element Descriptor

Bytes	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.

8.2.5.5 SMBus/I2C Mux Element Descriptor

The SMBus/I2C Mux Element Descriptor is shown in Figure 166 and is used to describe an SMBus/I2C multiplexor element that connects 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 166: SMBus/I2C Mux Element Descriptor

<table>
<tr><th>Bytes</th><th>Factory Default</th><th>Description</th></tr>
<tr><td>00</td><td>05h</td><td><b>Type:</b> 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><b>Revision:</b> 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><b>Length:</b> 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><b>SMBus/I2C Address Info:</b> This field indicates the SMBus/I2C address and whether or not ARP is supported.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:1</td><td><b>SMBus/I2C Address:</b> This field contains the 7-bit SMBus/I2C address. Refer to Figure 16 for requirements.</td></tr>
<tr><td>0</td><td><b>ARP Capable:</b> This bit is set to ‘1’ if SMBus ARP is supported, else it is cleared to ‘0’. Refer to Figure 16 for requirements.</td></tr>
</table></td></tr>
</table>

Figure 166: SMBus/I2C Mux Element Descriptor

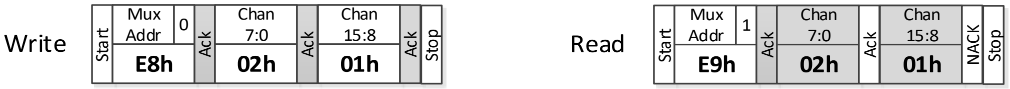
<table>
<tr><th>Bytes</th><th>Factory Default</th><th>Description</th></tr>
<tr><td>04</td><td>Impl Spec</td><td><b>SMBus/I2C Capabilities:</b> This field indicates the SMBus/I2C Mux capabilities.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7</td><td><b>Form Factor Reset:</b> This bit is set to ‘1’ if all of the SMBus/I2C reset mechanisms are supported as defined by the associated form factor specification. This bit 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><b>Packet Error Code (PEC) Support:</b> This bit is set to ‘1’ if PEC is supported by the SMBus/I2C Mux. This bit is cleared to ‘0’ if PEC is not supported.</td></tr>
<tr><td>5:2</td><td>Reserved</td></tr>
<tr><td>1:0</td><td><b>Maximum Speed:</b> 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><b>SMBus/I2C Mux Channel Descriptor Count:</b> 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>Impl Spec</td><td>Impl Spec</td><td><b>SMBus/I2C Mux Channel Descriptor 0:</b> This field contains the first SMBus/I2C Mux Channel Descriptor.</td></tr>
<tr><td>Impl Spec</td><td>Impl Spec</td><td><b>SMBus/I2C Mux Channel Descriptor 1:</b> This field contains the second SMBus/I2C Mux Channel Descriptor in this SMBus/I2C Mux Element Descriptor if SMBus/I2C Mux Channel Descriptor Count field 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 168. 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 166 may be truncated unless they are required 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 167 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 167: 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 SMBus/I2C Mux with 57 downstream channels would have 8 channel bytes. In the example shown in Figure 167, 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 0h. Bytes beyond the size required 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 (refer to Figure 146) or by an SMBus Reset (refer to section 8.3.4). Connecting or disconnecting channels while they are active is strongly discouraged and results in undefined behavior.

Figure 168: SMBus/I2C Mux Channel Descriptor

Bytes	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 field is not greater than one, then this field is not present.
…	…	…

8.2.5.6 PCIe Switch Element Descriptor

The PCIe Switch Element Descriptor is shown in Figure 169 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 169: PCIe Switch Element Descriptor

Bytes	Factory Default	Description
00	06h	**Type:** This field indicates the type of the Element Descriptor. The PCIe Switch Element Descriptor Type is 6h.
01	Impl Spec	**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 field 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 describes the downstream ports.

The format of a PCIe Switch Port Descriptor is shown in Figure 170. 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 170: PCIe Switch Port Descriptor

<table>
<tr><th>Bytes</th><th>Factory Default</th><th>Description</th></tr>
<tr><td>00</td><td>00h</td><td><b>Type:</b> This field indicates the type of Port Descriptor. The PCIe Switch Port Descriptor Type is 0.</td></tr>
<tr><td>01</td><td>Impl Spec</td><td><b>Length:</b> This field indicates the length of the PCIe Switch Port Descriptor in bytes.</td></tr>
<tr><td>02</td><td>Impl Spec</td><td><b>PCIe Link Speed:</b> This field indicates a bit vector of link speeds supported by the PCIe port.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:5</td><td>Reserved</td></tr>
<tr><td>4</td><td>Set to '1' if the PCIe link supports 32.0 GT/s, otherwise cleared to '0'.</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>
</table>

Figure 170: PCIe Switch Port Descriptor

<table>
<tr><th>Bytes</th><th>Factory Default</th><th>Description</th></tr>
<tr><td>03</td><td>Impl Spec</td><td><b>PCIe Maximum Link Width:</b> 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><b>RefClk Capability:</b> This field contains a bit vector that specifies the PCIe clocking modes supported by the port.
<table>
<tr><th>Bits</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><b>Port Number:</b> 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><b>PCIe Pointer:</b> 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><b>Destination Port:</b> 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>

8.2.5.7 NVM Subsystem Element Descriptor

The NVM Subsystem Element Descriptor is shown in Figure 171 and is used to describe an NVM Subsystem contained in the NVMe Storage Device.

Figure 171: NVM Subsystem Element Descriptor

Bytes	Factory Default	Description
00	07h	**Type:** This field indicates the type of the Element Descriptor. The NVM Subsystem Element Descriptor Type is 7h.
01	00h	**Revision:** This field indicates the revision of the Element Descriptor. The NVM Subsystem Element Descriptor Revision is 0h for this specification.
02	Impl Spec	**Length:** This field indicates the length of the NVM Subsystem Element Descriptor in bytes.

Figure 171: NVM Subsystem Element Descriptor

<table>
<tr><th>Bytes</th><th>Factory Default</th><th>Description</th></tr>
<tr><td>03</td><td>3Ah
or
3Bh</td><td><b>SMBus/I2C Address Info:</b> 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>Bits</th><th>Description</th></tr>
<tr><td>7:1</td><td><b>SMBus/I2C Address:</b> This field contains the 7-bit SMBus/I2C address. Refer to Figure 16 for requirements.</td></tr>
<tr><td>0</td><td><b>ARP Capable:</b> This bit is set to ‘1’ if SMBus ARP is supported, else it is cleared to ‘0’. Refer to Figure 16 for requirements.</td></tr>
</table></td></tr>
<tr><td>04</td><td>Impl Spec</td><td><b>SMBus/I2C Capabilities:</b> 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>Bits</th><th>Description</th></tr>
<tr><td>7</td><td><b>Reset:</b> This bit is set to ‘1’ if all of the SMBus/I2C reset mechanisms are supported as defined by the associated form factor specification. This bit 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><b>Maximum Speed:</b> 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><b>NVM Subsystem Port Descriptor Count:</b> 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><b>NVM Subsystem Port Descriptor 0:</b> 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><b>NVM Subsystem Port Descriptor 1:</b> This field contains the NVM Subsystem Port Descriptor associated with the second NVM Subsystem port if NVM Subsystem Port Descriptor Count field 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 172. 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 one parent Element Descriptor.

Figure 172: NVM Subsystem Port Descriptor

Bytes	Factory Default	Description
00	00h	**Type:** This field indicates the type of an NVM Subsystem Port Descriptor. The NVM Subsystem Port Descriptor Type is 0.
01	Impl Spec	**Length:** This field indicates the length of the NVM Subsystem Port Descriptor in bytes.

Figure 172: NVM Subsystem Port Descriptor

<table>
<tr><th>Bytes</th><th>Factory Default</th><th>Description</th></tr>
<tr><td>02</td><td>Impl Spec</td><td><b>PCIe Link Speed:</b> This field indicates a bit vector of link speeds supported by the PCIe port.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:5</td><td>Reserved</td></tr>
<tr><td>4</td><td>Set to ‘1’ if the PCIe link supports 32.0 GT/s, otherwise cleared to ‘0’.</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>03</td><td>Impl Spec</td><td><b>PCIe Maximum Link Width:</b> 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><b>RefClk Capability:</b> This field contains a bit vector that specifies the PCIe clocking modes supported by the port.
<table>
<tr><th>Bits</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><b>Port Identifier:</b> This field contains the NVMe-MI Port Identifier associated with this port.</td></tr>
</table>

8.2.5.8 FRU Information Device Element Descriptor

The FRU Information Device Element Descriptor is shown in Figure 173 and is used to describe a FRU Information Device contained in the NVMe Storage Device.

Figure 173: FRU Information Device Element Descriptor

Byte Offset	Factory Default	Description
00	08h	**Type:** This field indicates the type of the Element Descriptor. The FRU Information Device Element Descriptor Type is 8.
01	00h	**Revision:** This field indicates the revision of the Element Descriptor. The FRU Information Device Element Descriptor Revision is 0h for this specification.

Figure 173: FRU Information Device Element Descriptor

<table>
<tr><th>Byte Offset</th><th>Factory Default</th><th>Description</th></tr>
<tr><td>02</td><td>06h</td><td><b>Length:</b> This field indicates the length of the FRU Information Device Element Descriptor in bytes.</td></tr>
<tr><td>03</td><td>A6h/A7h or 0h for NVM Storage Devices A4h/A5h or 0h for Carriers</td><td><b>SMBus/I2C Address Info:</b> If the NVMe Storage Device contains an SMBus/I2C port, then this field indicates the default SMBus/I2C addressing per the table below; else, this field shall be cleared to 0h.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7:1</td><td><b>SMBus/I2C Address:</b> This field contains the 7-bit SMBus/I2C address. Refer to Figure 16 for requirements.</td></tr>
<tr><td>0</td><td><b>ARP Capable:</b> If this bit is set to ‘1’, then SMBus ARP is supported. If this bit is cleared to ‘0’, then SMBus ARP is not supported. Refer to the SMBus Specification for additional details.</td></tr>
</table></td></tr>
<tr><td>04</td><td>Impl Spec</td><td><b>SMBus/I2C Capabilities:</b> If the NVM Storage Device contains an SMBus/I2C port, then this field indicates the SMBus/I2C capabilities per the table below; else, this field shall be cleared to 0h.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7</td><td><b>Reset:</b> If this bit is set to ‘1’, then all of the SMBus/I2C reset mechanisms are supported as defined by the specification for the Form Factor in the Host Connector Element Descriptor.
If this bit is cleared to ‘0’, then the FRU Information 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</td><td><b>I2C Writes Allowed:</b> If this bit is set to ‘1’, then the FRU Information Device is allowed to be written using an I2C Write operation.
If this bit is cleared to ‘0’, then the FRU Information Device is not allowed to be written using an I2C Write operation.</td></tr>
<tr><td>5:2</td><td>Reserved</td></tr>
<tr><td>1:0</td><td><b>Maximum Speed</b>: This field is set to the highest supported SMBus/I2C clock speed supported by the FRU Information Device.
<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>8h to 0Ch inclusive</td><td><b>Maximum FRU Information Device Size:</b> The maximum size of the FRU Information Device is 2^N bytes where N is the value in this field (e.g., a value of 8 in this field indicates a maximum FRU Information Device size of 2^8 or 256 bytes).</td></tr>
</table>

8.2.5.9 Vendor-Specific Element Descriptors

The Vendor-Specific Element Descriptor is shown Figure 174.

Figure 174: Vendor-Specific Element Descriptors

Bytes	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 to 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.

Figure 174: Vendor-Specific Element Descriptors

Bytes	Factory Default	Description
04:03	Impl Spec	**PCI Vendor ID:** This field indicates PCI-SIG assigned vendor identifier.
Impl Spec	Impl Spec	**Vendor Specific:** Vendor-specific information.

8.3 Reset

This section describes the reset architecture defined by this specification that are applicable to NVMe Storage Devices and NVMe Enclosures.

8.3.1 NVM Subsystem Reset

An NVM Subsystem Reset is initiated under the conditions outlined in the NVM Express Base Specification (e.g., when main power is applied to the NVM Subsystem). In addition to these conditions, if NVM Subsystem Reset is supported, then it may be initiated by processing a Reset command.

An NVM Subsystem Reset initiated via the out-of-band mechanism may interfere with host software. A Management Controller should coordinate with the host. Coordination between a Management Controller and a host are outside the scope of this specification.

When an NVM Subsystem Reset is initiated, the entire NVM Subsystem is reset. This includes all NVM Subsystem ports (PCIe and SMBus/I2C), Management Endpoints, and Controller Management Interfaces. All state is returned to its default condition.

8.3.2 Controller Level Reset

A Controller Level Reset is initiated under the conditions outlined in the NVM Express Base Specification.

A Controller Level Reset initiated via the out-of-band mechanism may interfere with host software. A Management Controller should coordinate with the host. Coordination between a Management Controller and a host are outside the scope of this specification.

The actions performed on a Controller Level Reset are outlined in the NVM Express Base Specification. A Controller Level Reset has no effect on the Controller Management Interface associated with that Controller, the PCI Express port associated with that Controller, or a Management Endpoint associated with that port. A Controller Level Reset also does not stop the servicing of the Management Interface Command Set or NVM Express Admin Command Set commands that target that Controller (i.e., the NVM Express Admin Command Set is still available even though the NVMe Controller may be disabled or held in reset) or Control Primitives. A Controller Level Reset may affect PCIe Command Set commands being processed on that Controller (refer to section 8.1). If a PCIe Command is affected, then the command is completed with status PCIe Inaccessible.

A Controller Level Reset that causes a new firmware image to activate is considered a special event and may impact the operation of the Controller Management Interface associated with one or more Controllers, servicing of NVMe-MI Messages, or Management Endpoints within an NVM Subsystem. This impact is unspecified and vendor specific. The Management Controller and host should coordinate the activation of a new firmware image. Coordination between a Management Controller and a host are outside the scope of this specification.

8.3.3 Management Endpoint Reset

A Management Endpoint Reset is initiated under the conditions outlined in the MCTP Base Specification or the associated MCTP transport binding specifications.

In addition to these conditions, a Management Endpoint associated with a PCI Express port is reset when the PCI Express port is in a PCI Express conventional reset state.

When a Management Endpoint Reset is initiated, the state of that Management Endpoint is returned to its default condition and any commands associated with that Management Endpoint being processed are aborted. A reset of a Management Endpoint in an NVM Subsystem shall not affect any other Management Endpoint in the NVM Subsystem or any other NVM Subsystem entity. Note that for implementations compliant to version 1.1 and earlier of this specification, during a PCI Express conventional reset of a PCIe Management Endpoint, MCTP accesses may not be supported on other PCIe or SMBus/I2C Management Endpoints in the NVM Subsystem.

8.3.4 SMBus Reset

All SMBus/I2C elements should support the recommendation for SMBus Reset when the SMBus/I2C clock is low for longer than $t_{TIMEOUT,MIN}$ (refer to the SMBus Specification).

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

If the SMBus/I2C element on an NVM Subsystem is transmitting a Response Message, 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 NVM Subsystem shall remain idle on SMBus for the remainder of the SMBus Reset assertion even if other SMBus/I2C elements attempt to address it. An NVM Subsystem 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 are to be reinitialized.

An SMBus Reset shall be treated by each Command Slot in the SMBus/I2C Management Endpoint as if an implicit Abort Control Primitive (refer to section 4.2.1.3) was received with the exception that the Management Endpoint does not transmit the Abort Control Primitive Response Messages.

8.4 Security

The Responder may respond with a Response Message Status of Access Denied in an Error Response. While a drive is in an unlocked state, this mechanism shall not be used for the Management Interface Command Set or the NVMe Admin Command Set.

The commands and the times at which such a response is generated is vendor specific. The mechanism used to lock a drive is outside the scope of this specification.

Appendix A Technical Note: NVM Express Basic Management Command

This appendix describes the NVMe Basic Management Command and is included here for informational purposes only. The NVMe Basic Management Command is not formally a part of this specification and its features are not tested by the NVMe Compliance program. No further enhancements to the NVMe Basic Management Command are planned, and it is strongly recommended that any consumers of the NVMe Basic Management Command transition to using the standard NVMe-MI protocol.

This specification utilizes Management Component Transport Protocol (MCTP) messages. The NVMe Basic Management Command does not utilize MCTP. Support for the NVMe Basic Management Command is optional.

This command does not provide any mechanism to modify or configure the NVMe device. Modifying or configuring the NVMe device requires the more capable MCTP protocol rather than this command's SMBus Block Read. The host may reuse existing SMBus or FRU Information Device read subroutines for this read.

The block read protocol is specified by the SMBus Specification which is available online at www.smbus.org. First SMBus address write and command code bytes are transmitted by the host, then a repeated start and finally a SMBus address read. The host keeps clocking as the drive responds with the selected data. The command code is used as a starting offset into the data block shown in Figure 153, like an address on a serial EEPROM.

The offset value increments on every byte read and is reset to 0h on a stop condition. A read command without a repeated start is permissible and starts transmission from offset zero. Reading more than the block length with an I2C read is also permissible and these reads continue into the first byte in the next block of data. The Packet Error Code (PEC) accumulates all bytes sent or received after the start condition and the current value is inserted whenever a PEC field is reached.

Blocks of data are packed sequentially. The first 2 blocks are defined by the NVMe-MI workgroup. The first block is the dynamic host health data. The second block includes the Vendor ID (VID) and serial number of the drive. Additional blocks of data may be defined by the owner of the VID. Reading past the end of the vendor defined blocks shall return zeros.

The SMBus address to read this data structure defaults to D4h. After the Management Controller successfully assigns the MCTP UDID to D4h using ARP, then the Basic Management Command may track and respond to reads at future ARP assigned MCTP addresses. This method of changing the Basic Command address is optional and does not persist through power cycles. Interleaved MCTP and block read traffic is permissible and neither command type shall disturb the state of the other commands.

Here are a few example reads from an NVMe drive at 30 °C, no alarms, VID=1234h, serial number is AZ123456 using the format defined in Figure 153. Host transmissions are shown in white blocks and drive responses are shown in grey blocks:

Example 1: SMBus block read of the drive's status (status flags, SMART warnings, temperature):

Start	Addr	W	Ack	Cmd Code	Ack	Restart	Addr	R	Ack	Length	Ack	Status Flags	Ack	SMART Warnings	Ack	Temp	Ack	Drive Life Used	Ack	Warning Temp	Ack	Power State	Ack	PEC	NACK	Stop
	D4h			00h			D5h			06h		BFh		FFh		1Eh		01h		3Ch		08h		10h		

Example 2: SMBus block read of the drive’s static data (VID and serial number):

Start	Addr W	Ack	Cmd Code	Ack	Restart	Addr R	Ack	Length	Ack	VID	Ack	VID	Ack	Serial # ‘A’	Ack	Serial # ‘Z’	Ack	Serial # ‘1’	Ack	Serial # ‘2’	Ack	Serial # ‘3’	Ack	Serial # ‘4’	Ack
	D4h		08h			D5h		16h		12h		34h		41h		5Ah		31h		32h		33h		34h	

Serial # ‘5’	Ack	Serial # ‘6’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack
35h		36h		20h		20h		20h		20h		20h		20h		20h		20h		20h		20h	

Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	PEC	NACK	Stop
20h		20h		DAh		

Example 3: SMBus send byte to reset Arbitration bit:

Start	Addr W	Ack	Cmd Code	Ack	Stop
	D4h		FFh		

Example 4: I2C read of status and vendor content, I2C allows reading across SMBus block boundaries:

Start	Addr W	Ack	Cmd Code	Ack	Restart	Addr R	Ack	Length	Ack	Status Flags	Ack	SMART Warnings	Ack	Temp	Ack	Drive Life Used	Ack	Warning Temp	Ack	Power State	Ack	PEC	Ack	Length	Ack
	D4h		00h			D5h		06h		BFh		FFh		1Eh		01h		3Ch		08h		10h		16h	

VID	Ack	VID	Ack	Serial # ‘A’	Ack	Serial # ‘Z’	Ack	Serial # ‘1’	Ack	Serial # ‘2’	Ack	Serial # ‘3’	Ack	Serial # ‘4’	Ack	Serial # ‘5’	Ack	Serial # ‘6’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack
12h		34h		41h		5Ah		31h		32h		33h		34h		35h		36h		20h		20h	

Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	Serial # ‘ ’	Ack	PEC	NACK	Stop
20h		20h		20h		20h		20h		20h		20h		20h		20h		20h		B0h		

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

1. The host does an SMBus byte write to send byte FFh which clears the SMBus Arbitration bit on all listening Management Endpoints at this SMBus address;
2. The host does an I2C read starting from offset 0h and continuing at least through the serial number in the second block. The drive transmitting a ‘0’ when other drives sent a ‘1’ wins arbitration and sets the arbitration bit to ‘1’ upon read completion to give other drives priority on the next read;
3. Repeat step 2 until all drives are read, host receiving the Arbitration bit as a ‘1’ indicates loop is done; and
4. Sort the responses by serial number since the order of drive responses varies with health status and temperatures.

Be careful that there are no short reads of similar data between steps 1 and 3. If the read data is exactly the same on multiple drives, then all these drives set the arbitration bit. After that a new send byte FFh is required to restart the process.

The logic levels were intentionally inverted to normally high in the bytes 1 and 2. This is an additional mechanism to assist systems that do not have ARP or muxes. Since ‘0’ bits win arbitration on SMBus, a drive with an alarm condition is prioritized over healthy drives in the above arbitration scheme. A single I2C read of byte of two bytes starting at offset one from an array of drives detects alarm conditions. Note that only one drive with an alarm may be reliably detected because drives without the same alarm stop transmitting once the bus contention is detected. For this reason, the bits are sorted in order of priority. Continuing to read further provides the serial number of the drive that had the alarm.

Figure 175: Subsystem Management Data Structure

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 address. **Drive Not Ready** – Bit 6 is set to '1' when the NVMe 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. **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 require 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'.
	02	**SMART Warnings:** This field shall contain the Critical Warning field (byte 0) of the NVMe SMART / Health Information log. Each bit in this field shall be inverted from the NVMe definition (i.e., the management interface shall indicate a '0' value while the corresponding bit is '1' in the log page). Refer to the NVM Express Base Specification for bit definitions. If there are multiple Controllers in the NVM Subsystem, the Management Endpoint shall combine the Critical Warning field from every Controller such that a bit in this field is: • Cleared to '0' if any Controller in the NVMe Subsystem indicates a critical warning for that corresponding bit. • Set to '1' if all Controllers in the NVM Subsystem do not indicate a critical warning for the corresponding bit.

Figure 175: Subsystem Management Data Structure

<table>
<tr><th>Command Code</th><th>Offset (byte)</th><th>Description</th></tr>
<tr><td></td><td>03</td><td><b>Composite Temperature (CTemp):</b> This field indicates the current temperature in degrees Celsius. If a temperature value is reported, it should be the same temperature as the Composite Temperature from the SMART log of hottest Controller in the NVM Subsystem. The reported temperature range is vendor specific, and shall not exceed the range -60 °C to +127°C.
This field should not report a stale temperature, which means that it was sampled more than 5 s prior. If recent data is not available, the Management Endpoint should indicate a value of 80h for this field.
The field values are shown below.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>00h to 7Eh</td><td>Temperature is measured in degrees Celsius (0 °C to 126 °C)</td></tr>
<tr><td>7Fh</td><td>127 °C or higher</td></tr>
<tr><td>80h</td><td>No temperature data or temperature data is more the 5 s old.</td></tr>
<tr><td>81h</td><td>Temperature sensor failure</td></tr>
<tr><td>82h to C3h</td><td>Reserved</td></tr>
<tr><td>C4h</td><td>Temperature is -60 °C or lower</td></tr>
<tr><td>C5h to FFh</td><td>Temperature measured in degrees Celsius is represented in two's complement (-1 °C to -59 °C)</td></tr>
</table></td></tr>
<tr><td></td><td>04</td><td><b>Percentage Drive Life Used (PDLU):</b> Contains a vendor specific estimate of the percentage of NVM Subsystem NVM life used based on the actual usage and the manufacturer's prediction of NVM life. If an NVM Subsystem has multiple Controllers the highest value is returned. A value of 100 indicates that the estimated endurance of the NVM in the NVM Subsystem has been consumed but may not indicate an NVM Subsystem failure. The value is allowed to exceed 100. Percentages greater than 254 shall be represented as 255. This value should be updated once per power-on hour and equal the Percentage Used value in the NVMe SMART Health Log Page.</td></tr>
<tr><td></td><td>05</td><td><b>Current Over Temperature Warning Threshold (Optional):</b> This field indicates the composite temperature over temperature warning threshold in degrees Celsius. This is intended to initially match the temperature reported in the WCTEMP field in the NVMe Identify Controller data structure. If the Over Temperature threshold for Composite Temperature is modified with set features, then the most recent value should be reported. The data format should match the same single byte format as the CTemp field with a range from -60 C to 127 C. A value of 0h means that this field is not reported or that the threshold is set to 0 C.</td></tr>
<tr><td></td><td>06</td><td><b>Current Power (Optional):</b> This field reports the current NVM Subsystem power consumption. If both bit mapped fields are cleared to 0h, then this field is not reported.
<table>
<tr><th>Bit</th><th>Definition</th></tr>
<tr><td>7</td><td><b>NVM Subsystem Idle (NVMSI):</b> This bit is set to '1' when the NVM Subsystem is idle and has been idle for at least 5 s. Refer to the NVMe Idle Power (IDLP) definition.</td></tr>
<tr><td>6:0</td><td>NVM Subsystem Power (NVMSP): This field reports the ceiling function of the power consumed by the NVM Subsystem in Watts. If this value is greater than 127 W, then 127 W is reported.
Power reported by the NVM Subsystem is determined in the following manner. If NVMSI bit is set to '1', then the value returned is equal to that reported in the Idle Power (IDLP) field in the Power State Descriptor Data Structure for the corresponding NVMe power state. If NVMSI bit is cleared to '0', then the value returned is equal to that reported in the Active Power Workload (APW) field in the Power State Descriptor Structure for the corresponding NVMe power state. The Maximum Power (MP) field value is substituted for IDLP or APW if these are not for reported in the Power State Descriptor Structure for the current NVMe power state.</td></tr>
</table></td></tr>
<tr><td></td><td>07</td><td><b>PEC:</b> An 8 bit CRC calculated over the SMBus address, command code, second SMBus address, and returned data. The algorithm is defined in the SMBus Specification.</td></tr>
</table>

Figure 175: Subsystem Management Data Structure

Command Code	Offset (byte)	Description
8	08	**Length of identification:** Indicates number of additional bytes to read before encountering PEC. This value should always be 22 (16h) in implementations of this version of the spec.
	10:09	**Vendor ID:** The 2 byte vendor ID, assigned by the PCI-SIG. Should match VID in the Identify Controller command response. Note the MSB is transmitted first.
	11:30	**Serial Number:** 20 characters that match the serial number in the NVMe Identify Controller command response. Note the first character is transmitted first.
	31	**PEC:** An 8 bit CRC calculated over the SMBus address, command code, second SMBus address, and returned data. The algorithm is defined in the SMBus Specification.
32+	32:255	Vendor Specific – These data structures shall not exceed the maximum read length of 255 specified in the SMBus version 3 specification. Preferably their lengths are not greater than 32 for compatibility with SMBus 2.0.

Appendix B Example MCTP Messages & Message Integrity Check

Below are artificial MCTP Messages with their corresponding Message Integrity values. Figure 178 shows an example where the message is not an even number of dwords and the MIC spans Dwords 7 and 8. The contents of the messages listed below should be used for reference and do not correspond to valid MCTP messages.

Figure 176: MIC Example 1 – 32 Bytes of 0's

	3	2	1	0
Dword 0	00h	00h	00h	00h
`.	...	...	...	...
Dword 7	00h	00h	00h	00h
Dword 8 (MIC)	8Ah	91h	36h	AAh

Figure 177: MIC Example 2 – 32 Bytes of 1's

	3	2	1	0
Dword 0	FFh	FFh	FFh	FFh
...	...	...	...	...
Dword 7	FFh	FFh	FFh	FFh
Dword 8 (MIC)	62h	A8h	ABh	43h

Figure 178: MIC Example 3 – 30 Incrementing Bytes from 00h to 1Dh

	3	2	1	0
Dword 0	03h	02h	01h	00h
...	...	...	...	...
Dword 7 (MIC)	92h	D7h	1Dh	1Ch
Dword 8 (MIC)	<unused>		1Eh	05h

Figure 179: MIC Example 4 – 32 Decrementing Bytes from 1Fh to 00h

	3	2	1	0
Dword 0	1Ch	1Dh	1Eh	1Fh
...	...	...	...	...

Figure 179: MIC Example 4 – 32 Decrementing Bytes from 1Fh to 00h

	3	2	1	0
Dword 7	00h	01h	02h	03h
Dword 8 (MIC)	11h	3Fh	DBh	5Ch

Appendix C Example NVMe-MI Messages over SMBus/I2C

This section contains example NVMe-MI Messages over SMBus/I2C between a Management Controller (e.g., a Baseboard Management Controller) and a Management Endpoint. The Request Messages are sent from the Management Controller to the Management Endpoint and the corresponding Response Messages are sent back from the Management Endpoint to the Management Controller.

The examples assume the following:

- Management Endpoint SMBus/I2C address is 3Ah;
- Management Controller SMBus/I2C address is 20h;
- Management Endpoint MCTP Endpoint ID is 0, examples only use SMBus/I2C address;
- Management Controller MCTP Endpoint ID is 0, examples only use SMBus/I2C address;
- MCTP Transmission Unit Size is 64 bytes;
- NVMe Storage Device Composite Temperature (CTEMP) is 30 °C;
- NVMe Storage Device Controller ID is 1; and
- NVMe Storage Device Serial Number is AZ123456.

The first 4 bytes and the last byte of each packet (shown in orange in the examples below) are defined by the MCTP SMBus/I2C Transport Binding Specification. Bytes 4 to 7 of each packet and the Message Integrity Check (**green**) are defined by the MCTP Base Specification. The CRC-32C algorithm and the NVMe-MI Message Header (**blue**) are defined in section 3.1.1.1. Management Controller transmission bytes are shown in white blocks and Management Endpoint transmission bytes are shown in grey blocks. The MCTP endpoint sending the messages drives the clock pin so the signal direction changes between commands and responses as described in the MCTP binding specification.

Example 1: In this example, a Management Controller issues an Identify Command to read the Serial Number (bytes 23:04 of the Identify Controller Data Structure) of an NVMe Storage Device. The NVMe Storage Device's response is shown in the Example 2.

The Request Message is longer than the default 64-byte MCTP Transmission Unit Size and thus spans two MCTP packets. The NVMe-MI Message Type (NMIMT) field specifies that this is an NVMe Admin Command. The NVMe Opcode 06h specifies that this is an Identify Command. This NVMe Opcode and the required values for Dwords 1 to 15 are defined in the NVM Express Base Specification for the Identify Command. The Data Offset of 00000004h skips the first 4 bytes of the Identify Controller Data Structure response. The Data Length of 00000014h limits the response to 20 bytes.

Notice that the blue header is only present in the first packet of a message. The MCTP packet sequence number is incremented from 0 for the first packet to 1 for the second packet. The SMBus PEC is calculated per packet and includes every byte sent. The Message Integrity Check is calculated across both packet payloads but skips all orange and green bytes. The value for SMBus Length field (Byte 2) is the number of bytes following it in the packet, not including the SMBus PEC field per the SMBus Specification.

Start	SSD Addr	0	Ack	Protocol= MCTP	Ack	Length	Ack	BMC Addr	1	Ack	MCTP Version	Ack	SSD EID	Ack	BMC EID	Ack	flags,seq, own, tag	Ack	Type = NVMe-MI	Ack	NVMe Admin	Ack	Rsvd	Ack	Rsvd	Ack
	3Ah			0Fh		45h		21h			01h		00h		00h		8Bh		84h		10h		00h		00h	

Opcode= Identify	Ack	Flags= Len+ Off	Ack	Cntrl Id LSB	Ack	Cntrl Id MSB	Ack	Dword1 LSB	Ack	Dword1	Ack	Dword1	Ack	Dword1 MSB	Ack	Dword2 LSB	Ack	Dword2	Ack	Dword2	Ack	Dword2 MSB	Ack
06h		03h		01h		00h		00h		00h		00h		00h		00h		00h		00h		00h	

Dword3 LSB	Ack	Dword3	Ack	Dword3	Ack	Dword3 MSB	Ack	Dword4 LSB	Ack	Dword4	Ack	Dword4	Ack	Dword4 MSB	Ack	Dword5 LSB	Ack	Dword5	Ack	Dword5	Ack	Dword5 MSB	Ack
00h		00h		00h		00h		00h		00h		00h		00h		00h		00h		00h		00h	

Offset LSB	Ack	Offset	Ack	Offset	Ack	Offset MSB	Ack	Length LSB	Ack	Length	Ack	Length	Ack	Length MSB	Ack	Dword8 LSB	Ack	Dword8	Ack	Dword8	Ack	Dword8 MSB	Ack
04h		00h		00h		00h		14h		00h		00h		00h		00h		00h		00h		00h	

Dword9 LSB	Ack	Dword9	Ack	Dword9	Ack	Dword9 MSB	Ack	Dword10 LSB	Ack	Dword10	Ack	Dword10	Ack	Dword10 MSB	Ack	Dword11 LSB	Ack	Dword11	Ack	Dword11	Ack	Dword11 MSB	Ack
00h		00h		00h		00h		01h		00h		00h		00h		00h		00h		00h		00h	

Dword12 LSB	Ack	Dword12	Ack	Dword12	Ack	Dword12 MSB	Ack	Dword13 LSB	Ack	Dword13	Ack	Dword13	Ack	Dword13 MSB	Ack	Dword14 LSB	Ack	Dword14	Ack	Dword14	Ack	Dword14 MSB	Ack
00h		00h		00h		00h		00h		00h		00h		00h		00h		00h		00h		00h	

PEC	Ack	Stop
B2h		

Start	SSD Addr	0	Ack	Protocol= MCTP	Ack	Length	Ack	BMC Addr	1	Ack	MCTP Version	Ack	SSD EID	Ack	BMC EID	Ack	flags,seq, own, tag	Ack	Dword15 LSB	Ack	Dword15	Ack	Dword15	Ack	Dword15 MSB	Ack
	3Ah			0Fh		0Dh		21h			01h		00h		00h		5Bh		00h		00h		00h		00h	

CRC32C LSB	Ack	CRC32C	Ack	CRC32C	Ack	CRC32C MSB	Ack	PEC	Ack	Stop
4Ah		C3h		2Ch		FAh		EFh		

Example 2: This example shows an NVMe Storage Device's Response Message to the Identify Command from Example 1. This message is small enough to fit in a single packet so both MCTP SOM and EOM flags are set. The NVM Express Base Specification defines the format (Dwords 0, 1, and 3) of the Identify Controller Data Structure bytes that are returned.

Note that the SMBus/I2C addresses and MCTP Endpoint IDs in the Response Message are swapped from their order in the Request Message. Also note that the incrementing MCTP packet sequence number for the Management Endpoint is independent from the Management Controller's MCTP packet sequence number.

Start	BMC Addr	0	Ack	Protocol= MCTP	Ack	Length	Ack	SSD Addr	1	Ack	MCTP Version	Ack	BMC EID	Ack	SSD EID	Ack	flags, seq own, tag	Ack	Type= NVMe-MI	Ack	NVMe Admin	Ack	Rsvd	Ack	Rsvd	Ack
	20h			0Fh		31h		3Bh			01h		00h		00h		C3h		84h		90h		00h		00h	

Status= Success	Ack	Rsvd	Ack	Rsvd	Ack	Rsvd	Ack	Dword0 LSB	Ack	Dword0	Ack	Dword0	Ack	Dword0 MSB	Ack	Dword1 LSB	Ack	Dword1	Ack	Dword1	Ack	Dword1 MSB	Ack
00h		00h		00h		00h		00h		00h		00h		00h		00h		00h		00h		00h	

Dword3 LSB	Ack	Dword3	Ack	Dword3	Ack	Dword3 MSB	Ack	Response Data 'A'	Ack	Response Data 'Z'	Ack	Response Data '1'	Ack	Response Data '2'	Ack	Response Data '3'	Ack	Response Data '4'	Ack	Response Data '5'	Ack	Response Data '6'	Ack
00h		00h		00h		00h		41h		5Ah		31h		32h		33h		34h		35h		36h	

Response Data ' '	Ack	Response Data ' '	Ack	Response Data ' '	Ack	Response Data ' '	Ack	Response Data ' '	Ack	Response Data ' '	Ack	Response Data ' '	Ack	Response Data ' '	Ack	Response Data ' '	Ack	Response Data ' '	Ack	Response Data ' '	Ack	Response Data ' '	Ack
20h		20h		20h		20h		20h		20h		20h		20h		20h		20h		20h		20h	

CRC32C LSB	Ack	CRC32C	Ack	CRC32C	Ack	CRC32C MSB	Ack	PEC	Ack	Stop
7Ah		1Fh		C4h		7Bh		48h		

Example 3: In this example, a Management Controller issues an NVM Subsystem Health Status Poll command and clears the Composite Controller Status. Note that the MCTP packet sequence number is incremented from the last packet the Management Controller sent in Example 1. The NVMe-MI Message Type value of 08h with Opcode 01h makes this an NVM Subsystem Health Status Poll command. Bit 31 of Dword1 set to '1' clears the Composite Controller Status after preparing the response. Only the first non SR-IOV PCI function with any of the trigger able changes is requested.

Start	SSD Addr	0	Ack	Protocol= MCTP	Ack	Length	Ack	BMC Addr	1	Ack	MCTP Version	Ack	SSD EID	Ack	BMC EID	Ack	flags, seq own, tag	Ack	Type = NVMe-MI	Ack	Cmd = NVMe-MI	Ack	Rsvd	Ack	Rsvd	Ack
	3Ah			0Fh		19h		21h			01h		00h		00h		EBh		84h		08h		00h		00h	

Opcode= SubSys	Ack	Rsvd	Ack	Rsvd	Ack	Rsvd	Ack	Dword0 LSB	Ack	Dword0	Ack	Dword0	Ack	Dword0 MSB	Ack	Dword1 LSB	Ack	Dword1	Ack	Dword1	Ack	Dword1 MSB	Ack
01h		00h		00h		00h		00h		00h		00h		00h		00h		00h		00h		80h	

CRC32C LSB	Ack	CRC32C	Ack	CRC32C	Ack	CRC32C MSB	Ack	PEC	Ack	Stop
AAh		EFh		81h		B4h		48h		

Example 4: This example shows an NVMe Storage Device's response to the NVM Subsystem Health Status Poll command from Example 3. Note that the MCTP packet sequence number is incremented from the last packet the NVMe Storage Device sent in Example 2. Controller ID 0 had a reportable trigger due to its composite temperature change.

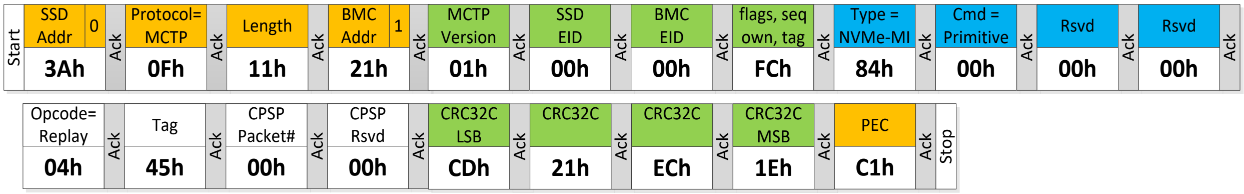
Start	BMC Addr	0	Ack	Protocol= MCTP	Ack	Length	Ack	SSD Addr	1	Ack	MCTP Version	Ack	BMC EID	Ack	SSD EID	Ack	flags, seq own, tag	Ack	Type= NVMe-MI	Ack	Cmd= NVMe-MI	Ack	Rsvd	Ack	Rsvd	Ack
	20h			0Fh		19h		3Bh			01h		00h		00h		D3h		84h		88h		00h		00h	

Status= Success	Ack	Rsvd	Ack	Rsvd	Ack	Rsvd	Ack	Subsystem Status	Ack	SMART Warnings	Ack	Composite Temp.	Ack	Percent Life Used	Ack	Ctlr Stat LSB	Ack	Ctlr Stat MSB	Ack	Rsvd	Ack	Rsvd	Ack
00h		00h		00h		00h		38h		FFh		1Eh		05h		01h		00h		00h		00h	

CRC32C LSB	Ack	CRC32C	Ack	CRC32C	Ack	CRC32C MSB	Ack	PEC	Ack	Stop
C8h		3Bh		3Bh		57h		DAh		

Example 5: This example shows a Management Controller issuing a Replay Control Primitive. The Management Controller may choose to replay an entire Response Message if, for example, the Message Integrity Check failed on the initial Response Message. Or the Management Controller may choose to replay a partial message starting at a specified MCTP Transmission Unit Size boundary if, for example, the

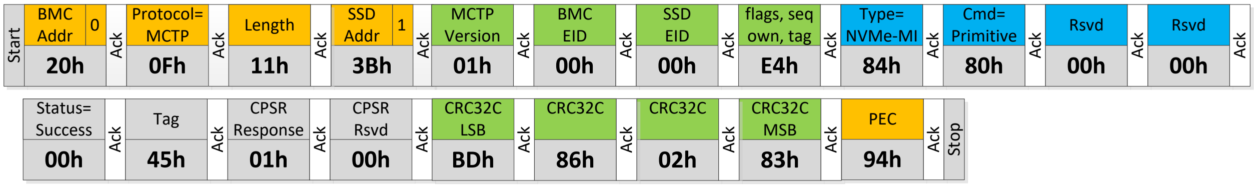
SMBus PEC failed on an individual packet. The Control Primitive Tag is arbitrarily set to 45h and remembered by the Management Controller to match response packets to the correct Control Primitives. The MCTP Tag is also modified for this example to show the effect on the replayed packet.



Start	SSD Addr	0	Ack	Protocol= MCTP	Ack	Length	Ack	BMC Addr	1	Ack	MCTP Version	Ack	SSD EID	Ack	BMC EID	Ack	flags, seq own, tag	Ack	Type = NVMe-MI	Ack	Cmd = Primitive	Ack	Rsvd	Ack	Rsvd	Ack
	3Ah			0Fh		11h		21h			01h		00h		00h		FCh		84h		00h		00h		00h	

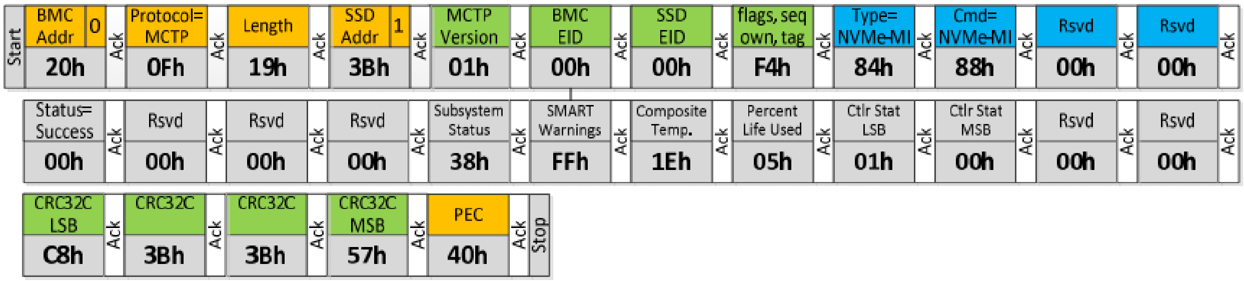
Opcode= Replay	Ack	Tag	Ack	CPSP Packet#	Ack	CPSP Rsvd	Ack	CRC32C LSB	Ack	CRC32C	Ack	CRC32C	Ack	CRC32C MSB	Ack	PEC	Ack	Stop
04h		45h		00h		00h		CDh		21h		ECh		1Eh		C1h		

Example 6: This example shows an NVMe Storage Device sending an acknowledgement Response Message to the Replay Control Primitive and then sending a second Response Message that replays the previous Response Message from specified offset of 0h. Note that the previous command is not reissued because that could return different data after having the Composite Controller Status cleared.



Start	BMC Addr	0	Ack	Protocol= MCTP	Ack	Length	Ack	SSD Addr	1	Ack	MCTP Version	Ack	BMC EID	Ack	SSD EID	Ack	flags, seq own, tag	Ack	Type= NVMe-MI	Ack	Cmd= Primitive	Ack	Rsvd	Ack	Rsvd	Ack
	20h			0Fh		11h		3Bh			01h		00h		00h		E4h		84h		80h		00h		00h	

Status= Success	Ack	Tag	Ack	CPSR Response	Ack	CPSR Rsvd	Ack	CRC32C LSB	Ack	CRC32C	Ack	CRC32C	Ack	CRC32C MSB	Ack	PEC	Ack	Stop
00h		45h		01h		00h		BDh		86h		02h		83h		94h		



Start	BMC Addr	0	Ack	Protocol= MCTP	Ack	Length	Ack	SSD Addr	1	Ack	MCTP Version	Ack	BMC EID	Ack	SSD EID	Ack	flags, seq own, tag	Ack	Type= NVMe-MI	Ack	Cmd= NVMe-MI	Ack	Rsvd	Ack	Rsvd	Ack
	20h			0Fh		19h		3Bh			01h		00h		00h		F4h		84h		88h		00h		00h	

Status= Success	Ack	Rsvd	Ack	Rsvd	Ack	Rsvd	Ack	Subsystem Status	Ack	SMART Warnings	Ack	Composite Temp.	Ack	Percent Life Used	Ack	Ctlr Stat LSB	Ack	Ctlr Stat MSB	Ack	Rsvd	Ack	Rsvd	Ack
00h		00h		00h		00h		38h		FFh		1Eh		05h		01h		00h		00h		00h	

CRC32C LSB	Ack	CRC32C	Ack	CRC32C	Ack	CRC32C MSB	Ack	PEC	Ack	Stop
C8h		3Bh		3Bh		57h		40h		