NVM Express®

Base Specification

Revision 2.0c

October 4th, 2022

Please send comments to info@nvmexpress.org

NVM Express® Base Specification, Revision 2.0c is available for download at https://nvmexpress.org. The NVM Express Base Specification, Revision 2.0c incorporates NVM Express Base Specification, Revision 2.0 (refer to https://nvmexpress.org/changes-in-nvm-express-revision-2-0 for details), ECN 001, ECN102, ECN105, ECN106, ECN107, ECN109, and ECN110.

SPECIFICATION DISCLAIMER

LEGAL NOTICE:

This NVM Express Base Specification, Revision 2.0c 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 Base Specification, Revision 2.0c 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: **"© 2008 to 2022 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.
InfiniBand™ is a trademark and servicemark of the InfiniBand Trade Association.

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

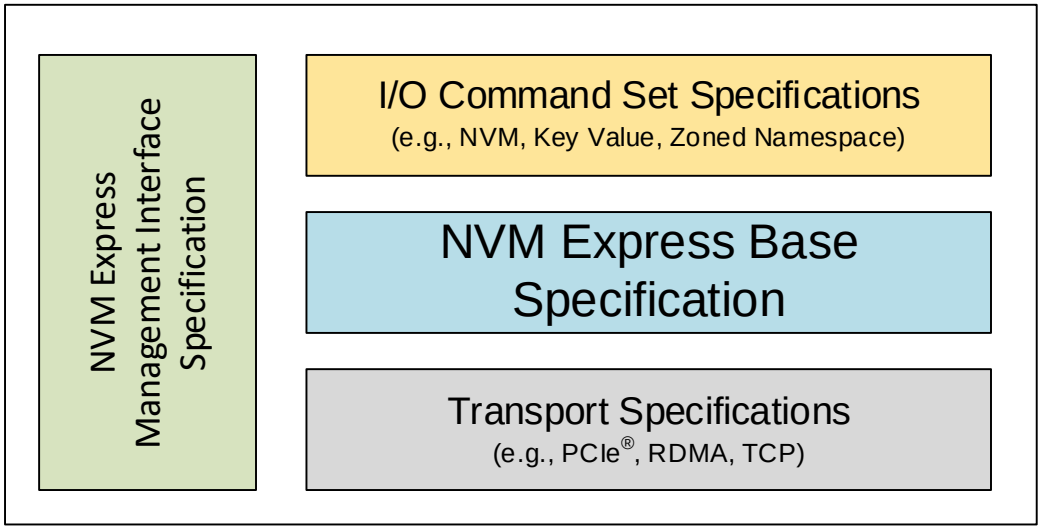
The NVM Express® (NVMe®) interface allows host software to communicate with a non-volatile memory subsystem. This interface is optimized for all storage solutions, attached using a variety of transports including PCI Express®, Ethernet, InfiniBand™, and Fibre Channel. The mapping of extensions defined in this document to a specific NVMe Transport are defined in an NVMe Transport binding specification. The NVMe Transport binding specification for Fibre Channel is defined in INCITS 556 Fibre Channel – Non-Volatile Memory Express - 2 (FC-NVMe-2).

For an overview of changes from revision 1.4 to revision 2.0, refer to http://nvmexpress.org/changes for a document that describes the new features, including mandatory requirements for a controller to comply with revision 2.0.

1.1.1 NVM Express® Specification Family

Figure 1 shows the relationship of the NVM Express specifications to each other within the NVMe™ family of specifications.

Figure 1: NVMe Family of Specifications



The NVM Express specification family structure shown in Figure 1 is intended to show the applicability of NVM Express specifications to each other, not a hierarchy, protocol stack, or system architecture.

The NVM Express Base (NVM Express Base) Specification (i.e., this specification) defines a protocol for host software to communicate with a non-volatile memory subsystems over a variety of memory-based transports and message-based transports.

The NVM Express Management Interface (NVMe-MI) Specification defines an optional management interface for all NVM Express Subsystems.

NVM Express I/O Command Set specifications define data structures, features, log pages, commands, and status values that extend the NVM Express Base Specification.

NVM Express Transport specifications define the binding of the NVMe protocol including controller properties to a specific transport.

1.2 Scope

This specification defines a set of properties and commands that comprise the interface required for communication with a controller in an NVM subsystem. These properties are to be implemented by an instance of a controller using a specific NVMe Transport. This specification also defines common aspects of the NVMe I/O Command Sets that may be supported by a controller.

There are three types of controllers with different capabilities (refer to section 3.1.2):

a) I/O controllers;
b) Discovery controllers; and
c) Administrative controllers.

In this document the generic term controller is often used instead of enumerating specific controller types when applicable controller types may be determined from the context.

1.3 Outside of Scope

The property interface and command set are specified apart from any usage model for the NVM, but rather only specifies the communication interface to the NVM subsystem. Thus, this specification does not specify whether the non-volatile memory system is used as a solid state drive, a main memory, a cache memory, a backup memory, a redundant memory, etc. Specific usage models are outside the scope, optional, and not licensed.

This specification defines requirements and behaviors that are implementation agnostic. The implementation of these requirements and behaviors are outside the scope of this specification. For example, an NVM subsystem that follows this specification may be implemented by an SSD that attaches directly to a fabric, a device that translates between a fabric and a PCIe NVMe SSD, or software running on a general purpose server.

This interface is specified above any non-volatile memory management, like wear leveling. Erases and other management tasks for NVM technologies like NAND are abstracted.

This specification does not contain any information on caching algorithms or techniques.

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, PCI Express, and PCI-X). This includes published specifications for fabrics and other technologies referred to by this document or any NVMe Transport binding specification.

1.4 Conventions

1.4.1 Keywords

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

1.4.1.1 mandatory

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

1.4.1.2 may

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

1.4.1.3 obsolete

A keyword indicating functionality that was defined in a previous version of the NVM Express specification and that has been removed from this specification.

1.4.1.4 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.

1.4.1.5 R

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

1.4.1.6 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, property, or register shall be cleared to 0h, or in accordance with a future extension to this specification. The recipient of a command or a register write is not required to check reserved bits, bytes, words, or fields. Receipt of reserved coded values in defined fields in commands shall be reported as an error. Writing a reserved coded value into a controller property field produces undefined results.

1.4.1.7 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.

1.4.1.8 should

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

1.4.2 Numerical Descriptions

A 0’s based value is a numbering scheme in which the number 0h represents a value of 1h, 1h represents 2h, 2h represents 3h, etc. In this numbering scheme, there is no method to represent 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.

Size values are shown in binary units or decimal units. The symbols used to represent these values are as shown in Figure 2.

Figure 2: 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}

The ^ operator is used to denote the power to which that number, symbol, or expression is to be raised.

Some parameters are defined as an ASCII string. ASCII strings shall contain only code values 20h through 7Eh. For the string “Copyright”, the character “C” is the first 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 hexadecimal ASCII string is an ASCII string that uses a subset of the code values: “0” to “9”, “A” to “F” uppercase, and “a” to “f” lowercase.

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).

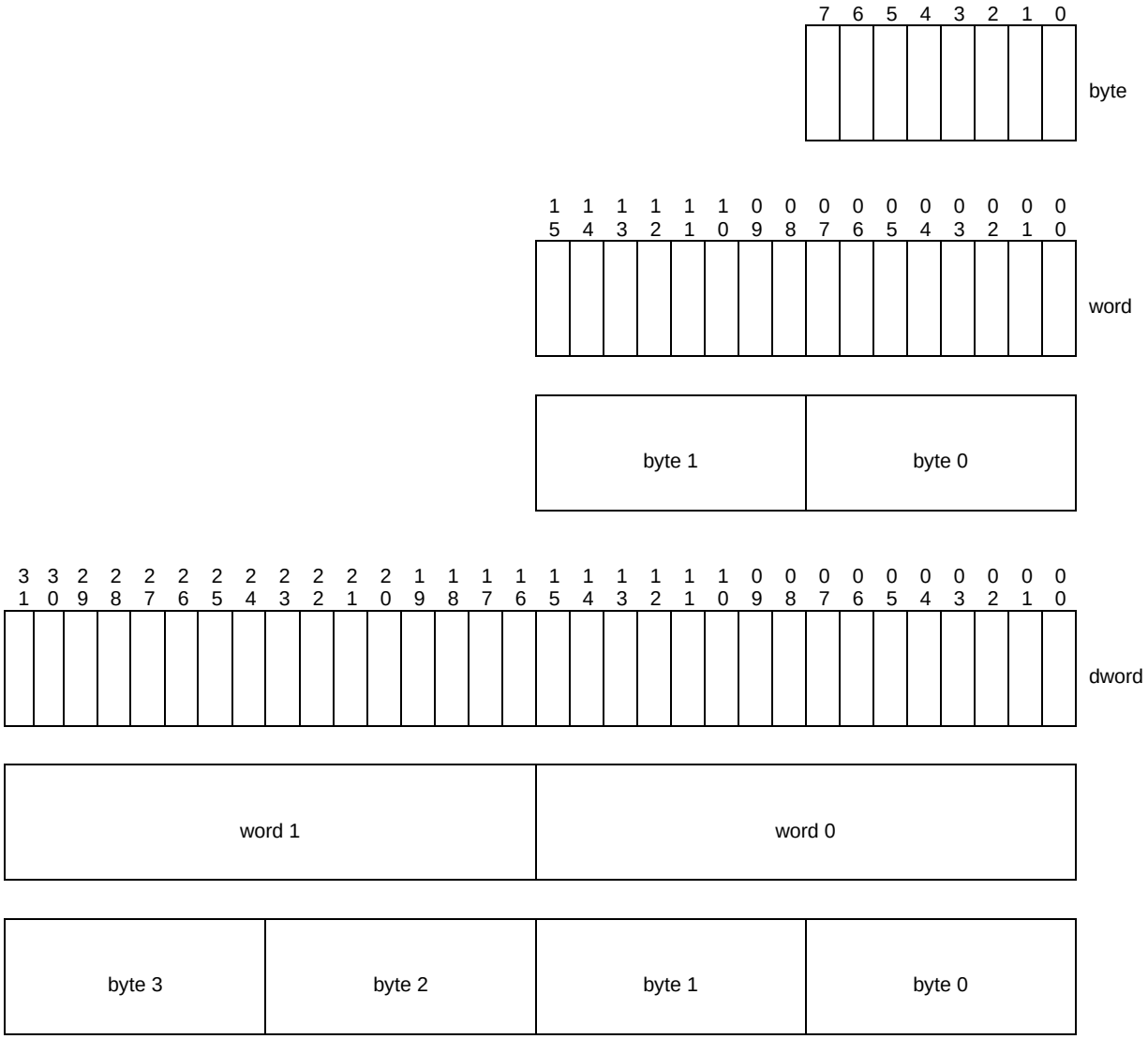
All other numbers are decimal (i.e., base 10). A decimal number is represented in this specification by any sequence of digits consisting of only the Western-Arabic numerals 0 to 9 not immediately followed by a lower-case b or a lower-case h (e.g., 175). This specification uses the following conventions 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., 2019 instead of 2,019).

1.4.3 Byte, Word, and Dword Relationships

Figure 3 illustrates the relationship between bytes, words and dwords. A qword (quadruple word) is a unit of data that is four times the size of a word; it is not illustrated due to space constraints. Unless otherwise specified, this specification specifies data in a little endian format.

Figure 3: Byte, Word, and Dword Relationships



1.5 Definitions

1.5.1 Admin Queue

The Admin Queue is the Submission Queue and Completion Queue with identifier 0. The Admin Submission Queue and corresponding Admin Completion Queue are used to submit administrative commands and receive completions for those administrative commands, respectively.

The Admin Submission Queue is uniquely associated with the Admin Completion Queue.

1.5.2 Administrative controller

A controller that exposes capabilities that allow a host to manage an NVM subsystem. An Administrative controller does not implement I/O Queues, provide access to data or metadata associated with user data on a non-volatile memory storage medium, or support namespaces attached to the Administrative controller (i.e., there are never any active NSIDs).

1.5.3 arbitration burst

The maximum number of commands that may be fetched by an arbitration mechanism at one time from a Submission Queue.

1.5.4 arbitration mechanism

The method used to determine which Submission Queue is selected next to fetch commands for execution by the controller. Refer to section 3.4.4.

1.5.5 association

An exclusive communication relationship between a particular controller and a particular host that encompasses the Admin Queue and all I/O Queues of that controller.

1.5.6 audit

The process of accessing media to determine correct operation of a sanitize operation. Refer to section 8.21 and to ISO/IEC 27040.

1.5.7 authentication commands

Used to refer to Fabrics Authentication Send or Authentication Receive commands.

1.5.8 cache

A data storage area used by the NVM subsystem, that is not accessible to a host, and that may contain a subset of user data stored in the non-volatile media or may contain user data that is not committed to non-volatile media.

1.5.9 candidate command

A candidate command is a submitted command which has been transferred into the controller and the controller deems ready for processing.

1.5.10 capsule

An NVMe unit of information exchange used in NVMe over Fabrics. A capsule contains a command or response and may optionally contain command/response data and SGLs.

1.5.11 Channel

A Channel represents a communication path between the controller and one or more Media Units in an NVM subsystem.

1.5.12 command completion

A command is completed when the controller has completed processing the command, has updated status information in the completion queue entry, and has posted the completion queue entry to the associated Completion Queue.

1.5.13 command submission

For memory-based transport model (e.g. PCIe) implementations, a command is submitted when a Submission Queue Tail Doorbell write has completed that moves the Submission Queue Tail Pointer value past the Submission Queue slot in which the command was placed.

For message-based transport model (e.g. NVMe over Fabrics) implementations, a command is submitted when a host adds a capsule to a Submission Queue.

1.5.14 controller

A controller is the interface between a host and an NVM subsystem. There are three types of controllers:

a) I/O controllers;
b) Discovery controllers; and
c) Administrative controllers.

A controller executes commands submitted by a host on a Submission Queue and posts a completion on a Completion Queue. All controllers implement one Admin Submission Queue and one Admin Completion Queue. Depending on the controller type, a controller may also implement one or more I/O Submission Queues and I/O Completion Queues. When PCI Express is used as the transport, then a controller is a PCI Express function.

1.5.15 directive

A method of information exchange between a host and either an NVM subsystem or a controller. Information may be transmitted using the Directive Send and Directive Receive commands. A subset of I/O commands may include a Directive Type field and a Directive Specific field to communicate more information that is specific to the associated I/O command. Refer to section 8.7.

1.5.16 Discovery controller

A controller that exposes capabilities that allow a host to retrieve a Discovery Log Page. A Discovery controller does not implement I/O Queues or provide access to a non-volatile memory storage medium. Refer to section 3.1.2.3.

1.5.17 Discovery Service

An NVM subsystem that supports Discovery controllers only. A Discovery Service shall not support a controller that exposes namespaces.

1.5.18 dynamic controller

The controller is allocated on demand with no state (e.g., Feature settings) preserved from prior associations.

1.5.19 Domain

A domain is the smallest indivisible unit that shares state (e.g., power state, capacity information).

1.5.20 emulated controller

An NVM Express controller that is defined in software. An emulated controller may or may not have an underlying physical NVMe controller (e.g., physical PCIe function).

1.5.21 Endurance Group

A portion of NVM in the NVM subsystem whose endurance is managed as a group. Refer to section 3.2.3.

1.5.22 fabric (network fabric)

A network topology in which nodes pass data to each other.

1.5.23 firmware/boot partition image update command sequence

The sequence of one or more Firmware Image Download commands that download a firmware image or a boot partition image followed by a Firmware Commit command that commits that downloaded image to a firmware slot or a boot partition.

1.5.24 firmware slot

A firmware slot is a location in a domain used to store a firmware image. The domain stores from one to seven firmware images. Controllers in the same domain share the same firmware slots.

1.5.25 host

An entity that interfaces to an NVM subsystem through one or more controllers and submits commands to Submission Queues and retrieves command completions from Completion Queues.

1.5.26 host-accessible memory

Memory that the host is able to access (e.g., host memory, Controller Memory Buffer (CMB), Persistent Memory Region (PMR)).

1.5.27 host memory

Memory that may be read and written by both a host and a controller and that is not exposed by a controller (i.e., Controller Memory Buffer or Persistent Memory Region). Host memory may be implemented inside or outside a host (e.g., a memory region exposed by a device that is neither the host nor controller).

1.5.28 Identify Controller data structures

All controller data structures that are able to be retrieved via the Identify command: Identify Controller data structure (i.e., CNS 01h) and each of the I/O Command Set specific Identify Controller data structure (i.e., CNS 06h).

1.5.29 Identify Namespace data structures

All namespace data structures that are able to be retrieved via the Identify command: Identify Namespace data structure (i.e., CNS 00h), I/O Command Set Independent Identify Namespace data structure (i.e., CNS 08h), and each of the I/O Command Set specific Identify Namespace data structures (i.e., 05h).

1.5.30 I/O command

An I/O command is a command submitted to an I/O Submission Queue.

1.5.31 I/O Completion Queue

An I/O Completion Queue is a Completion Queue that is used to indicate command completions and is associated with one or more I/O Submission Queues.

1.5.32 I/O controller

A controller that implements I/O queues and is intended to be used to access a non-volatile memory storage medium.

1.5.33 I/O Submission Queue

An I/O Submission Queue is a Submission Queue that is used to submit I/O commands for execution by the controller (e.g., Read command and Write command for the NVM Command Set).

1.5.34 Media Unit

A Media Unit represents a component of the underlying media in an NVM subsystem. Endurance Groups are composed of Media Units.

1.5.35 metadata

Metadata is contextual information related to formatted user data (e.g., a particular LBA of data). The host may include metadata to be stored by the NVM subsystem if storage space is provided by the controller. Refer to the applicable I/O Command Set specification for details.

1.5.36 namespace

A formatted quantity of non-volatile memory that may be directly accessed by a host.

1.5.37 Namespace ID (NSID)

An identifier used by a controller to provide access to a namespace or the name of the field in the SQE that contains the namespace identifier (refer to Figure 87). Refer to section 3.2.1 for the definitions of valid NSID, invalid NSID, active NSID, inactive NSID, allocated NSID, and unallocated NSID.

1.5.38 NVM

NVM is an acronym for non-volatile memory.

1.5.39 NVM Set

A portion of NVM from an Endurance Group. Refer to section 3.2.2.

1.5.40 NVM subsystem

An NVM subsystem includes one or more domains, one or more controllers, zero or more namespaces, and one or more ports. An NVM subsystem may include a non-volatile memory storage medium and an interface between the controller(s) in the NVM subsystem and non-volatile memory storage medium.

1.5.41 NVM subsystem port

An NVMe over Fabrics protocol interface between an NVM subsystem and a fabric. An NVM subsystem port is a collection of one or more physical fabric interfaces that together act as a single interface.

1.5.42 NVMe over Fabrics

An implemenation of the NVM Express interface that complies to either the message-only or the message/memory-based implementation of the memory-based transport model definition (refer to Figure 4 and section 2.2).

1.5.43 NVMe Transport

A protocol layer that provides reliable delivery of data, commands, and responses between a host and an NVM subsystem. The NVMe Transport layer is layered on top of the fabric. It is independent of the fabric physical interconnect and low level fabric protocol layers.

1.5.44 NVMe Transport binding specification

A specification of reliable delivery of data, commands, and responses between a host and an NVM subsystem for an NVMe Transport. The binding may exclude or restrict functionality based on the NVMe Transport's capabilities.

1.5.45 physical fabric interface (physical ports)

A physical connection between an NVM subsystem and a fabric.

1.5.46 Port ID

An identifier that is associated with an NVM subsystem port. Refer to section 2.2.2.

1.5.47 primary controller

An NVM Express controller that supports the Virtualization Management command. An NVM subsystem may contain multiple primary controllers. Secondary controller(s) in an NVM subsystem depend on a primary controller for dynamic resource management (refer to section 8.26).

A PCI Express SR-IOV Physical Function that supports the NVM Express interface and the Virtualization Enhancements capability is an example of a primary controller (refer to section 8.26.4).

1.5.48 private namespace

A namespace that is only able to be attached to one controller at a time. Refer to the Namespace Multi-path I/O and Namespace Sharing Capabilities (NMIC) field in Figure 280.

1.5.49 property

The generalization of memory mapped controller registers defined for NVMe over PCIe. Properties are used to configure low level controller attributes and obtain low level controller status.

1.5.50 rotational media

Media that stores data on rotating platters (refer to section 8.20).

1.5.51 Runtime D3 (Power Removed)

In Runtime D3 (RTD3) main power is removed from the controller. Auxiliary power may or may not be provided. For PCI Express, RTD3 is the $D3_{cold}$ power state (refer to section 8.15.4).

1.5.52 sanitize operation

Process by which all user data in the NVM subsystem is altered such that recovery of the previous user data from any cache or the non-volatile media is infeasible for a given level of effort (refer to ISO/IEC 27040).

1.5.53 secondary controller

An NVM Express controller that depends on a primary controller in an NVM subsystem for management of some controller resources (refer to section 8.26).

A PCI Express SR-IOV Virtual Function that supports the NVM Express interface and receives resources from a primary controller is an example of a secondary controller (refer to section 8.26.4).

1.5.54 shared namespace

A namespace that may be attached to two or more controllers in an NVM subsystem concurrently. Refer to the Namespace Multi-path I/O and Namespace Sharing Capabilities (NMIC) field in Figure 280.

1.5.55 spindown

The process of changing a spindle from an operational power state to a non-operational power state, for an Endurance Group that stores data on rotational media (refer to section 8.20).

1.5.56 spinup

The process of changing a spindle from a non-operational power state to an operational power state, for an Endurance Group associated with rotational media (refer to section 8.20).

1.5.57 static controller

The controller is pre-existing with a specific Controller ID and its state (e.g., Feature settings) is preserved from prior associations.

1.5.58 user data

Data stored in a namespace that is composed of data that the host may store and later retrieve including metadata if supported.

1.6 I/O Command Set specific definitions used in this specification

The following terms used in this specification are defined in each I/O Command Set specification.

1.6.1 Endurance Group Host Read Command

An I/O Command Set specific command that results in the controller reading user data, but may or may not return the data to the host.

1.6.2 Format Index

A value used to index into the I/O Command Set Specific Format table (i.e., the User Data Format number).

1.6.3 SMART Data Units Read Command

An I/O Command Set specific command that results in the controller reading user data, but may or may not return the data to the host.

1.6.4 SMART Host Read Command

An I/O Command Set specific command that results in the controller reading user data, but may or may not return the data to the host.

1.6.5 User Data Format

An I/O Command Set specific format that describes the layout of the data on the NVM media.

1.6.6 User Data Out Command

An I/O Command Set specific command that results in the controller writing user data, but may or may not transfer user data from the host to the controller.

1.7 NVM Command Set specific definitions used in this specification

The following terms used in this specification are defined in the NVM Command Set Specification. These terms are used throughout the document as examples for a specific I/O Command Set.

1.7.1 logical block

The smallest addressable data unit for Read and Write commands.

1.7.2 logical block address (LBA)

The address of a logical block, referred to commonly as LBA.

1.8 References

INCITS 502-2019, Information technology – SCSI Primary Commands - 5 (SPC-5). Available from http://webstore.ansi.org.

INCITS 514-2014, Information technology – SCSI Block Commands - 3 (SBC-3). Available from http://webstore.ansi.org.

INCITS 529-2018, Information technology – ATA/ATAPI Command Set - 4 (ACS-4). Available from http://webstore.ansi.org.

INCITS 556-2020, Information technology – Non-Volatile Memory Express - 2 (FC-NVMe-2). Available from http://webstore.ansi.org.

ISO 8601, Data elements and interchange formats – Information interchange – Representations of dates and times. Available from https://www.iso.org.

ISO/IEC 27040:2015 Information technology – Security techniques – Storage security. Available from https://www.iso.org.

JEDEC JESD218B-01: Solid State Drive (SSD) Requirements and Endurance Test Method standard. Available from https://www.jedec.org.

NVM Express Management Interface Specification, Revision 1.2. Available from https://www.nvmexpress.org.

NVM Express NVM Command Set 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.

NVM Express Key Value 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 RDMA Transport Specification, Revision 1.0. Available from https://www.nvmexpress.org

NVM Express TCP Transport Specification, Revision 1.0. Available from https://www.nvmexpress.org.

PCI Local Bus Specification, revision 3.0. Available from https://www.pcisig.com.

PCI Express® Base Specification, Revision 4.0. Available from https://www.pcisig.com.

PCI Bus Power Management Interface Specification Revision 1.2. Available from https://www.pcisig.com.

PCI Single Root I/O Virtualization and Sharing Specification, revision 1.1. Available from https://www.pcisig.com/specifications/iov/single_root/.

PCI Firmware Specification Revision 3.2. Available from https://www.pcisig.com.

PCI Code and ID Assignment Specification Revision 1.11, 24 January, 2019. Available from https://www.pcisig.com.

RFC 1952, P. Deutsch, “GZIP file format specification version 4.3”, May 1996. Available from https://www.ietf.org/rfc.html.

RFC 1994, W. Simpson, “PPP Challenge Handshake Authentication Protocol (CHAP)”, August 1996. Available from https://www.ietf.org/rfc.html.

RFC 2104, H. Krawczyk, M. Bellare, R. Canetti, “HMAC: Keyed-Hashing for Message Authentication”, February 1997. Available from https://www.ietf.org/rfc.html.

RFC 2631, E. Rescorla, “Diffie-Hellman Key Agreement Method”, June 1999. Available from https://www.ietf.org/rfc.html.

RFC 3629, Alis Technologies, F. Yergeau, “UTF-8, a transformation format of ISO 10646”, November 2003. Available from https://www.ietf.org/rfc.html.

RFC 3986, T. Berners-Lee, R. Fielding, L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax”, January 2005. Available from https://www.ietf.org/rfc.html.

RFC 4086, D. Eastlake 3rd, J. Schiller, S. Crocker, “Randomness Requirements for Security”, June 2005. Available from https://www.ietf.org/rfc.html.

RFC 4122, P. Leach, M. Mealling, and R. Salz, “A Universally Unique Identifier (UUID) URN Namespace”, July 2005. Available from https://www.ietf.org/rfc.html.

RFC 4301, S. Kent, K. Seo, “Security Architecture for the Internet Protocol”, December 2005. Available from https://www.ietf.org/rfc.html.

RFC 4648, S. Josefsson, “The Base16, Base32, and Base64 Data Encodings”, October 2006. Available from https://www.ietf.org/rfc.html.

RFC 6234, D. Eastlake 3rd, and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", May 2011. Available from https://www.ietf.org/rfc.html.

RFC 7296, C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, T. Kivinen, “Internet Key Exchange Protocol Version 2 (IKEv2)”, October 2014. Available from https://www.ietf.org/rfc.html.

RFC 7919, D. Gillmor, “Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)”, August 2016. Available from https://www.ietf.org/rfc.html.

RFC 8446, E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.3”, August 2018. Available from https://www.ietf.org/rfc.html.

UEFI Specification Version 2.7A, September 2017. Available from https://uefi.org.

Advanced Configuration and Power Interface (ACPI) Specification, Version 6.2 Errata A, September 2017. Available from https://www.uefi.org.

TCG Storage Architecture Core Specification, Version 2.01 Revision 1.00. Available from https://www.trustedcomputinggroup.org.

TCG Storage Interface Interactions Specification (SIIS), Version 1.08 Revision 1.00. Available from https://www.trustedcomputinggroup.org.

1.9 References Under Development

INCITS 506-201x, SCSI Block Commands - 4 (SBC-4). Available from https://www.t10.org.

2 Theory of Operation

The NVM Express scalable interface is designed to address the needs of storage systems that utilize PCI Express based solid state drives or fabric connected devices. The interface provides optimized command submission and completion paths. It includes support for parallel operation by supporting up to 65,535 I/O Queues with up to 64 Ki - 1 outstanding commands per I/O Queue. Additionally, support has been added for many Enterprise capabilities like end-to-end data protection (compatible with SCSI Protection Information, commonly known as T10 DIF, and SNIA DIX standards), enhanced error reporting, and virtualization.

The interface has the following key attributes:

- Does not require uncacheable / MMIO register reads in the command submission or completion path;
- A maximum of one MMIO register write or one 64B message is necessary in the command submission path;
- Support for up to 65,535 I/O Queues, with each I/O Queue supporting up to 65,535 outstanding commands;
- Priority associated with each I/O Queue with well-defined arbitration mechanism;
- All information to complete a 4 KiB read request is included in the 64B command itself, ensuring efficient small I/O operation;
- Efficient and streamlined command set;
- Support for MSI/MSI-X and interrupt aggregation;
- Support for multiple namespaces;
- Efficient support for I/O virtualization architectures like SR-IOV;
- Robust error reporting and management capabilities; and
- Support for multi-path I/O and namespace sharing.

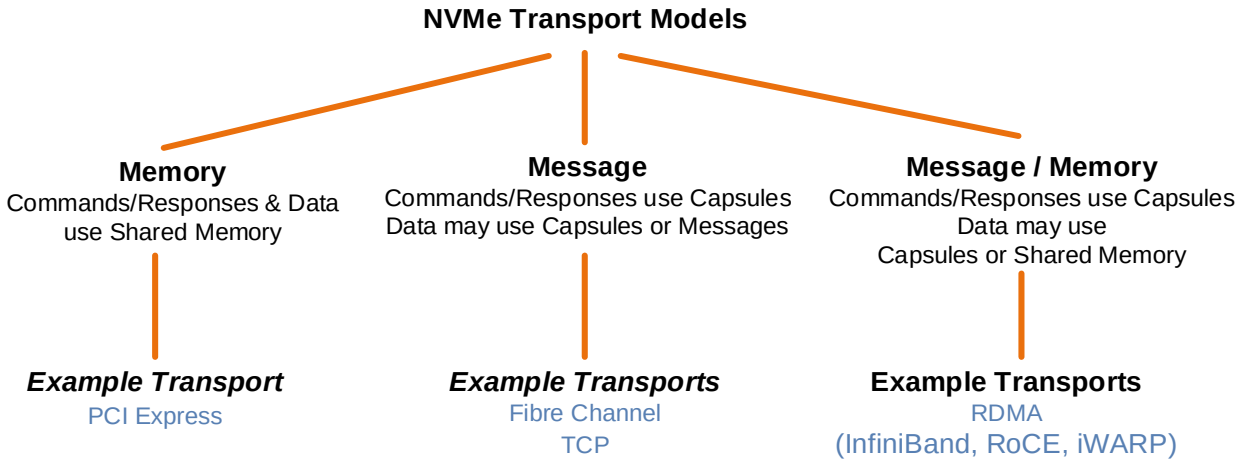
This specification defines a streamlined set of properties that are used to configure low level controller attributes and obtain low level controller status. These properties have a transport specific mechanism for defining access (e.g. Memory Mapped items use registers, whereas Fabrics use the Property Get and Property Set commands). The following are examples of functionality defined in properties:

- Indication of controller capabilities;
- Status for controller failures (command status is processed via CQ directly);
- Admin Queue configuration (I/O Queue configuration processed via Admin commands); and
- Doorbell registers for a scalable number of Submission and Completion Queues.

There are two defined constructs for communication between the host and the NVM subsystem, a memory-based transport model and a message-based transport model. All NVM subsystems require the underlying NVMe Transport to provide reliable NVMe command and data delivery. An NVMe Transport is an abstract protocol layer independent of any physical interconnect properties. A taxonomy of NVMe Transports along with examples is shown in Figure 4. An NVMe Transport may expose a memory-based transport model or a message-based transport model. The message-based transport model has two subtypes: the message-only transport model and the message/memory transport model. A memory-based transport model is one in which commands, responses, and data are transferred between a host and an NVM subsystem by performing explicit memory read and write operations. A message-based transport model is one in which messages containing command capsules and response capsules are sent between a host and an NVM subsystem. The two subtypes of message-based transport models are differentiated by how data is sent between a host and an NVM subsystem. In the message-only transport model data is only sent between a host and an NVM subsystem using capsules or messages. The message/memory-based transport model uses a combination of messages and explicit memory read and write operations to transfer command capsules, response capsules and data between a host and an NVM subsystem. Data may optionally be included in command capsules and response capsules. Both the message-only transport model and the message/memory-based transport model are referenced as message-based transport models throughout this specification when the description is applicable to both subtypes.

An NVM subsystem is made up of a single domain or multiple domains as described in section 3.2.4. An NVM subsystem may optionally include a non-volatile storage medium, and an interface between the controller(s) of the NVM subsystem and the non-volatile storage medium. Controllers expose this non-volatile storage medium to hosts through namespaces. An NVM subsystem is not required to have the same namespaces attached to all controllers. An NVM subsystem that supports a Discovery controller does not support any other controller type. A Discovery Service is an NVM subsystem that supports Discovery controllers only (refer to section 3.1).

Figure 4: Taxonomy of Transport Models



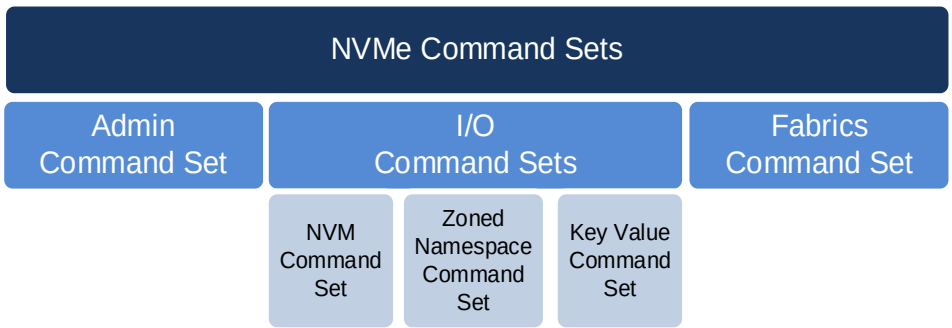
The capabilities and settings that apply to an NVM Express controller are indicated in the Controller Capabilities (CAP) property and the Identify Controller data structure (refer to Figure 275).

A namespace is a formatted quantity of non-volatile memory that may be accessed by a host. Associated with each namespace is an I/O Command Set that operates on that namespace. An NVM Express controller may support multiple namespaces that are referenced using a namespace ID. Namespaces may be created and deleted using the Namespace Management command and Capacity Management command. The Identify Namespace data structures (refer to section 1.5.29) indicate capabilities and settings that are specific to a particular namespace.

The NVM Express interface is based on a paired Submission and Completion Queue mechanism. Commands are placed by host software into a Submission Queue. Completions are placed into the associated Completion Queue by the controller.

There are three types of commands that are defined in NVM Express: Admin Commands, I/O Commands, and Fabrics Commands. Figure 5 shows these different command types.

Figure 5: Types of NVMe Command Sets



An Admin Submission Queue and associated Completion Queue exist for the purpose of controller management and control (e.g., creation and deletion of I/O Submission and Completion Queues, aborting commands, etc.). Only commands that are part of the Admin Command Set or the Fabrics Command Set may be submitted to the Admin Submission Queue.

An I/O Command Set is used with an I/O queue pair. This specification defines common I/O commands. I/O Command Sets are defined in NVMe I/O Command Set specifications (e.g., NVM Command Set, Key Value Command Set, or Zoned Namespace Command Set).

The Fabrics Command Set is NVMe over Fabrics specific. Fabrics Command Set commands are used for operations specific to NVMe over Fabrics including establishing a connection, NVMe in-band authentication, and to get or set a property. All Fabrics commands may be submitted on the Admin Submission Queue and some Fabrics commands may also be submitted on an I/O Submission Queue. Unlike Admin and I/O commands, Fabrics commands are processed by a controller regardless of whether the controller is enabled (i.e., regardless of the state of CC.EN).

2.1 Memory-Based Transport Model

In the memory-based model, Submission and Completion Queues are allocated in memory.

Host software creates queues, up to the maximum supported by the controller. Typically, the number of command queues created is based on the system configuration and anticipated workload. For example, on a four core processor based system, there may be a queue pair per core to avoid locking and ensure data structures are created in the appropriate processor core's cache. Figure 6 provides a graphical representation of the queue pair mechanism, showing a 1:1 mapping between Submission Queues and Completion Queues. Figure 7 shows an example where multiple I/O Submission Queues utilize the same I/O Completion Queue on Core B. Figure 6 and Figure 7 show that there is always a 1:1 mapping between the Admin Submission Queue and Admin Completion Queue.

Figure 6: Queue Pair Example, 1:1 Mapping

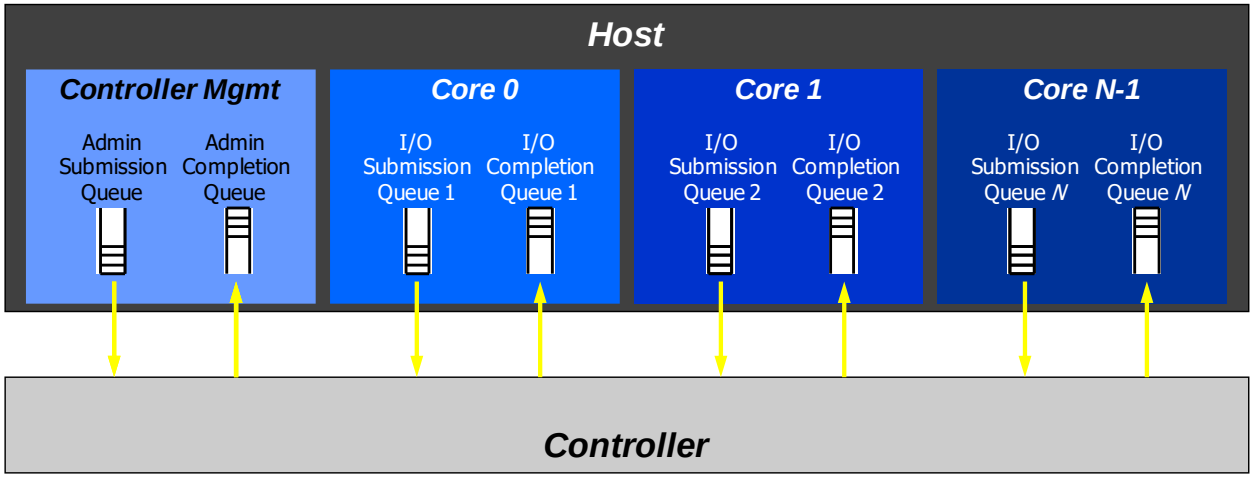
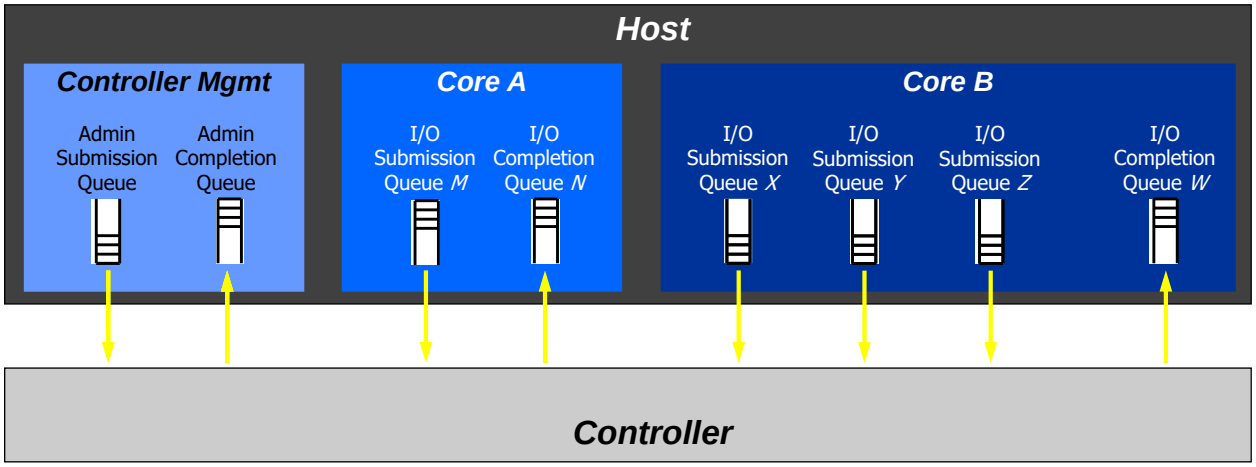


Figure 7: Queue Pair Example, *n*:1 Mapping



A Submission Queue (SQ) is a circular buffer with a fixed slot size that the host software uses to submit commands for execution by the controller. The host software updates the appropriate SQ Tail doorbell register when there are one to *n* new commands to execute. The previous SQ Tail value is overwritten in the controller when there is a new doorbell register write. The controller fetches SQ entries in order from the Submission Queue and may execute those commands in any order.

Each submission queue entry is a command. Commands are 64 bytes in size. The physical memory locations in memory to use for data transfers are specified using Physical Region Page (PRP) entries or Scatter Gather Lists (SGL). Each command may include two PRP entries or one Scatter Gather List segment. If more than two PRP entries are necessary to describe the data buffer, then a pointer to a PRP List that describes a list of PRP entries is provided. If more than one SGL segment is necessary to describe the data buffer, then the SGL segment provides a pointer to the next SGL segment.

A Completion Queue (CQ) is a circular buffer with a fixed slot size used to post status for completed commands. A completed command is uniquely identified by a combination of the associated SQ identifier and command identifier that is assigned by host software. In the memory-based transport model multiple Submission Queues may be associated with a single Completion Queue. A configuration with a single Completion Queue may be used where a single worker thread processes all command completions via one Completion Queue even when those commands originated from multiple Submission Queues. The CQ Head pointer is updated by host software after processing completion queue entries indicating the last free CQ slot. A Phase Tag (P) bit is defined in the completion queue entry to indicate whether an entry has been newly posted without the host consulting a register (refer to section 3.3.3.2.2). The Phase Tag bit enables the host to determine whether entries are new or not.

2.2 Message-Based Transport Model

The message-based transport model used for NVMe over Fabrics has the following differences from the memory-based transport model:

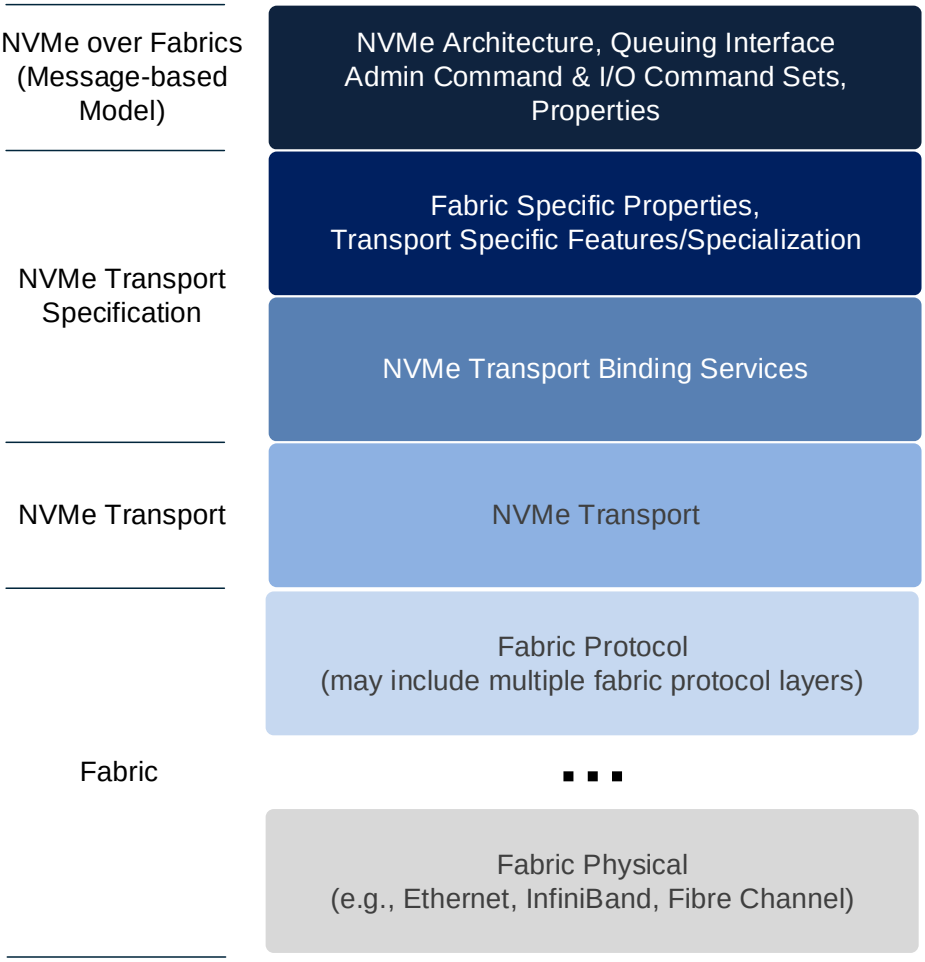
- There is a one-to-one mapping between I/O Submission Queues and I/O Completion Queues. NVMe over Fabrics does not support multiple I/O Submission Queues being mapped to a single I/O Completion Queue;
- NVMe over Fabrics does not define an interrupt mechanism that allows a controller to generate a host interrupt. It is the responsibility of the host fabric interface (e.g., Host Bus Adapter) to generate host interrupts;
- NVMe over Fabrics uses different mechanisms for I/O Submission Queue and I/O Completion Queue creation and deletion (refer to section 3.5);
- NVMe over Fabrics does not support transferring metadata from a separate buffer (e.g. does not support the Metadata Pointer field, refer to Figure 87);

- NVMe over Fabrics does not support PRPs but requires use of SGLs for Admin, I/O, and Fabrics commands. This differs from the memory-based transport model where SGLs are not supported for Admin commands and are optional for I/O commands;
- NVMe over Fabrics does not support Completion Queue flow control (refer to section 3.3.1.2.1). This requires that the host ensures there are available Completion Queue slots before submitting new commands; and
- NVMe over Fabrics allows Submission Queue flow control to be disabled if the host and controller agree to disable it. If Submission Queue flow control is disabled, the host is required to ensure that there are available Submission Queue slots before submitting new commands.

2.2.1 Fabrics and Transports

NVMe over Fabrics utilizes the protocol layering shown in Figure 8. This specification defines core aspects of the architecture that are independent of the NVMe Transport. An NVMe Transport binding specification is used to describe any NVMe Transport specific specialization as well as how the services required by the NVMe interface are mapped onto the corresponding NVMe Transport. The native fabric communication services and other functionality used by the NVMe interface and NVMe Transports (e.g., the Fabric Protocol and Fabric Physical layers in Figure 8) are outside the scope of the NVMe family of specifications.

Figure 8: NVMe over Fabrics Layering



2.2.2 NVM Subsystem Ports

An NVM subsystem presents a collection of one to (64Ki - 16) controllers which are used to access namespaces. The controllers may be associated with hosts through one to 64Ki NVM subsystem ports.

An NVM subsystem port is a protocol interface between an NVM subsystem and a fabric. An NVM subsystem port is a collection of one or more physical fabric interfaces that together act as a single protocol interface. When link aggregation (e.g., Ethernet) is used, the physical ports for the group of aggregated links constitute a single NVM subsystem port.

An NVM subsystem contains one or more NVM subsystem ports.

Each NVM subsystem port has a 16-bit port identifier (Port ID). An NVM subsystem port is identified by the NVM Subsystem NVMe Qualified Name (NQN) and Port ID. The NVM subsystem ports of an NVM subsystem may support different NVMe Transports. An NVM subsystem port may support multiple NVMe Transports if more than one NVMe Transport binding specifications exist for the underlying fabric (e.g., an NVM subsystem port identified by a Port ID may support both iWARP and RoCE). An NVM subsystem implementation may bind specific controllers to specific NVM subsystem ports or allow the flexible allocation of controllers between NVM subsystem ports, however, once connected, each specific controller is bound to a single NVM subsystem port.

A controller is associated with exactly one host at a time. NVMe over Fabrics allows multiple hosts to connect to different controllers in the NVM subsystem through the same NVM sybsystem port. All other aspects of NVMe over Fabrics multi-path I/O and namespace sharing (refer to section 2.4.1) are equivalent to that of the memory-based transport model.

2.2.3 Discovery Service

NVMe over Fabrics defines a discovery mechanism that a host uses to determine the NVM subsystems that expose namespaces that the host may access. The Discovery Service provides a host with the following capabilities:

- The ability to discover a list of NVM subsystems with namespaces that are accessible to the host;
- The ability to discover multiple paths to an NVM subsystem;
- The ability to discover controllers that are statically configured;
- The optional ability to establish explicit persistent connections to the Discovery controller; and
- The optional ability to receive Asynchronous Event Notifications from the Discovery controller.

A Discovery Service is an NVM subsystem that supports only Discovery controllers (refer to section 3.1.2.3), and shall not support any other controller type.

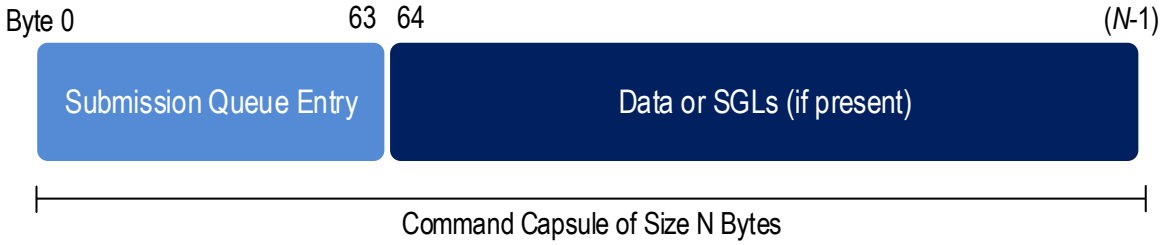
The method that a host uses to obtain the information necessary to connect to the initial Discovery Service is implementation specific. This information may be determined using a host configuration file, a hypervisor or OS property, or some other mechanism.

2.2.4 Capsules and Data Transfer

A capsule is an NVMe unit of information exchange used in NVMe over Fabrics. A capsule may be classified as a command capsule or a response capsule. A command capsule contains a command (formatted as a submission queue entry) and may optionally include SGLs or data. A response capsule contains a response (formatted as a completion queue entry) and may optionally include data. Data refers to any data transferred at an NVMe layer between a host and an NVM subsystem (e.g., logical block data or a data structure associated with a command). A capsule is independent of any underlying NVMe Transport unit (e.g., packet, message, or frame and associated headers and footers) and may consist of multiple such units.

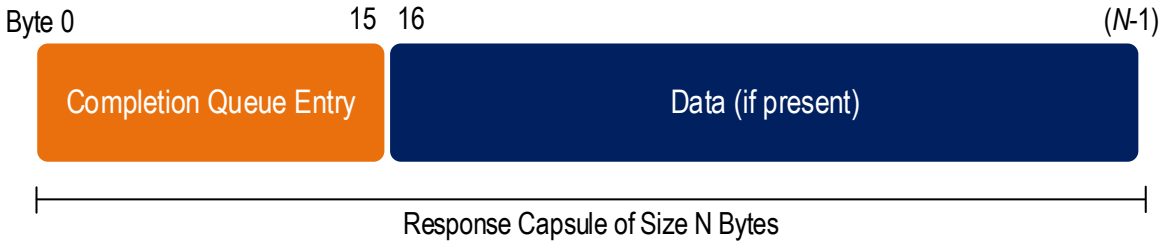
Command capsules are transferred from a host to an NVM subsystem. The SQE contains an Admin command, an I/O command, or a Fabrics command. The minimum size of a command capsule is NVMe Transport binding specific, but shall be at least 64B in size. The maximum size of a command capsule is NVMe Transport binding specific. The format of a command capsule is shown in Figure 9.

Figure 9: Command Capsule Format



Response capsules are transferred from an NVM subsystem to a host. The CQE is associated with a previously issued Admin command, I/O command, or Fabrics command. The size of a response capsule is NVMe Transport binding specific, but shall be at least 16B in size. The maximum size of a response capsule is NVMe Transport binding specific. The format of a response capsule is shown in Figure 10.

Figure 10: Response Capsule Format



NVMe Transports using the message-only transport model and message/memory transport model require all SGLs sent from the host to the controller be transferred within the command. They may optionally support the transfer of a portion or all data within the command and response capsules.

NVMe over Fabrics requires SGLs for all commands (Fabrics, Admin, and I/O). An SGL may specify the placement of data within a capsule or the information required to transfer data using an NVMe Transport specific data transfer mechanism (e.g., via memory transfers as in RDMA). Each NVMe Transport binding specification defines the SGLs used by a particular NVMe Transport and any capsule SGL and data placement restrictions.

2.2.5 Authentication

NVMe over Fabrics supports both fabric secure channel that includes authentication (refer to section 8.13.1) and NVMe in-band authentication. An NVM subsystem may require a host to use fabric secure channel, NVMe in-band authentication, or both. The Discovery Service indicates if fabric secure channel shall be used for an NVM subsystem. The Connect response indicates if NVMe in-band authentication shall be used with that controller.

A controller associated with an NVM subsystem that requires a fabric secure channel shall not accept any commands (i.e., Fabrics commands, Admin commands, or I/O commands) on an NVMe Transport until a secure channel is established. Following a Connect command, a controller that requires NVMe in-band authentication shall not accept any commands on the queue created by that Connect command other than authentication commands until NVMe in-band authentication has completed. Refer to section 8.13.

2.3 NVM Storage Model

2.3.1 Storage Entities

The NVM storage model includes the following entities:

- NVM subsystems (refer to 1.5.40)
- Domains (refer to section 3.2.4)

- Endurance Groups (refer to section 3.2.3)
- NVM Sets (refer to section 3.2.2)
- Namespaces (refer to section 3.2.1)

As illustrated below, each domain is contained in a single NVM subsystem, each Endurance Group is contained in a single domain, each NVM Set is contained in a single Endurance Group, and each namespace is contained in a single NVM Set. Each Media Unit is contained in a single Endurance Group.

Each Endurance Group is composed of storage media, which are termed Media Units (refer to section 8.3.2). For clarity, Media Units are not shown in the examples that follow.

Figure 11 shows the hierarchical relationships of these entities within a simple NVM subsystem, which has one domain, one Endurance Group, one NVM Set, and one namespace:

Figure 11: NVM Storage Hierarchy

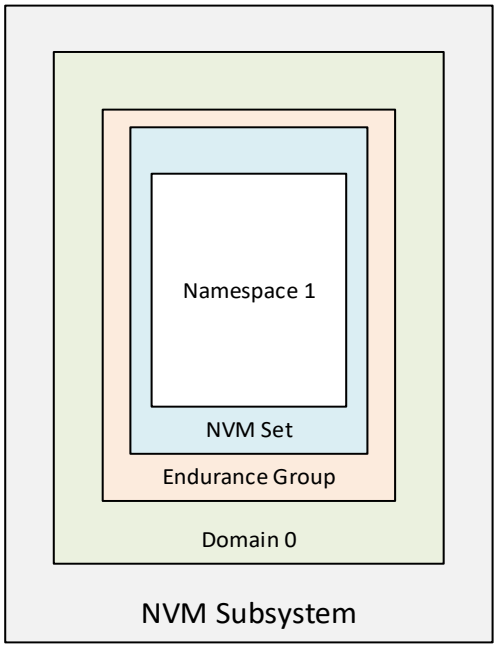
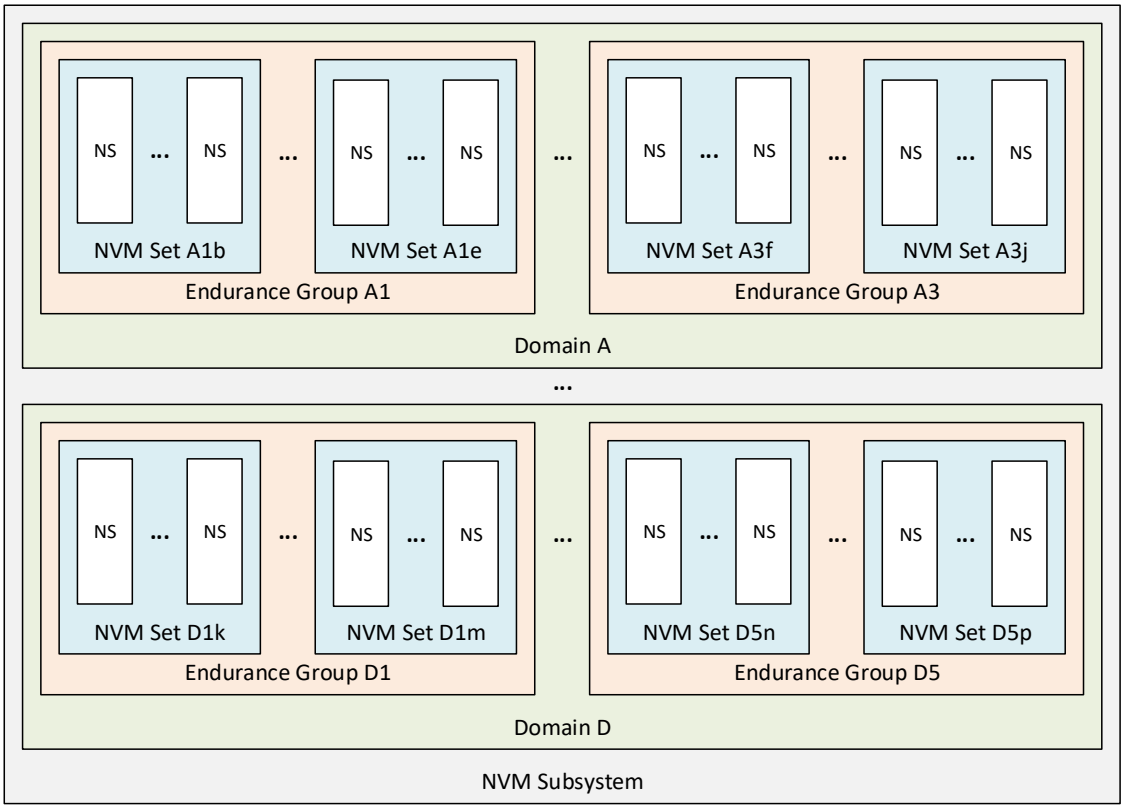


Figure 12 shows the relationships of these entities in a complex NVM subsystem, which has multiple domains, multiple Endurance Groups per domain, multiple NVM Sets per Endurance Group, and multiple namespaces per NVM Set:

Figure 12: Complex NVM Storage Hierarchy



Entity naming key (Abc):

- A: Domain (capital letter)
- b: Endurance Group (digit)
- c: NVM Set (lower case letter)

Reporting of Endurance Groups or NVM Sets is optional, but the storage model supports these concepts. An NVM subsystem may be shipped by the vendor with its storage entities configured, or it may be configured or re-configured by the customer. Typical changes to the configuration are creation and deletion of namespaces.

An NVM subsystem that does not support multiple NVM Sets does not require reporting of NVM Sets. An NVM subsystem that does not support multiple Endurance Groups does not require reporting of Endurance Groups.

2.3.2 I/O Command Sets

I/O commands perform operations on namespaces, and each namespace is associated with exactly one I/O command set. For example, commands in the NVM Command Set access data represented in a namespace as logical blocks, and commands in the Key Value Command Set access data represented in a namespace as key-value pairs.

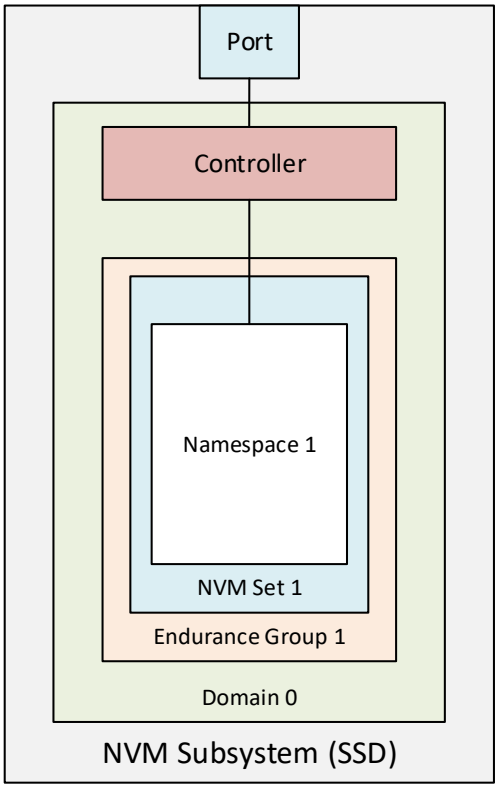
The association of a namespace to an I/O command set is specified when the namespace is created and is fixed for the lifetime of the namespace.

A controller may support one or more I/O command sets and may be attached to namespaces associated with different I/O command sets. A host issues commands to a namespace and those commands are interpreted based on the I/O command set associated with that namespace.

2.3.3 NVM Subsystem Examples

Figure 13 illustrates a simple NVM subsystem that has a single instance of each storage entity.

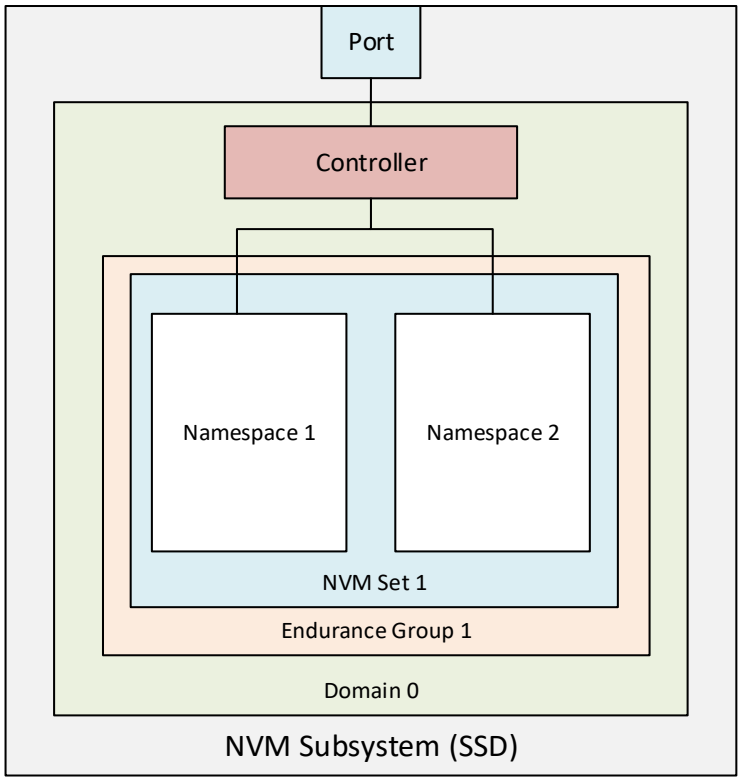
Figure 13: Single-Namespace NVM Subsystem



- The NVM subsystem consists of a single port and a single domain.
- The domain contains a controller and storage media.
- All of the storage media are contained in one Endurance Group.
- All of the storage media in that Endurance Group are organized into one NVM Set.
- That NVM Set contains a single namespace.

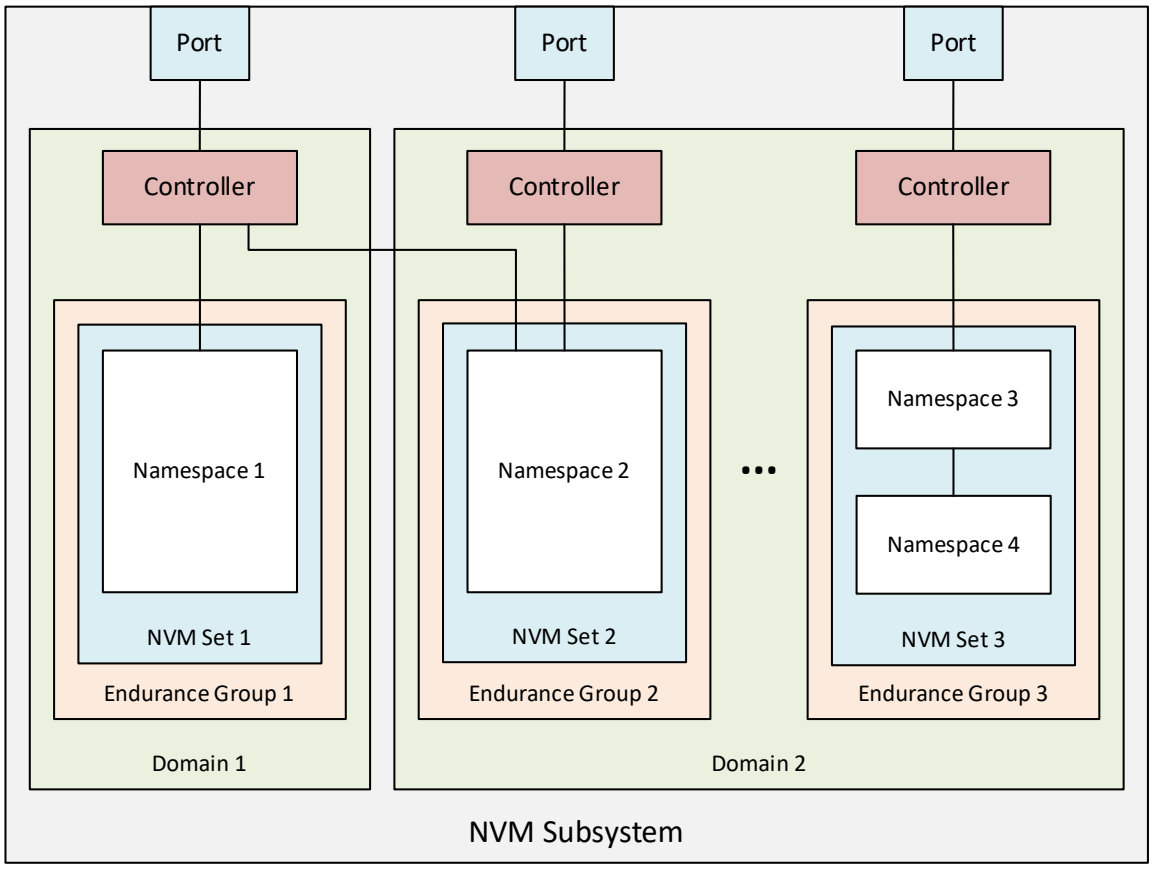
Figure 14 shows an NVM subsystem with two namespaces.

Figure 14: Two-Namespace NVM Subsystem



An NVM subsystem may have multiple domains, multiple namespaces, multiple controllers, and multiple ports, as shown in Figure 15.

Figure 15: Complex NVM Subsystem



2.4 Extended Capabilities Theory

2.4.1 Multi-Path I/O and Namespace Sharing

This section provides an overview of multi-path I/O and namespace sharing. Multi-path I/O refers to two or more completely independent paths between a single host and a namespace while namespace sharing refers to the ability for two or more hosts to access a common shared namespace using different NVM Express controllers. Both multi-path I/O and namespace sharing require that the NVM subsystem contain two or more controllers. NVM subsystems that support Multi-Path I/O and Namespace Sharing may also support asymmetric controller behavior (refer to section 2.4.2). Concurrent access to a shared namespace by two or more hosts requires some form of coordination between hosts. The procedure used to coordinate these hosts is outside the scope of this specification.

Figure 16 shows an NVM subsystem that contains a single NVM Express controller implemented over PCI Express and a single PCI Express port. Since this is a single Function PCI Express device, the NVM Express controller shall be associated with PCI Function 0. A controller may support multiple namespaces. The controller in Figure 16 supports two namespaces labeled NS A and NS B. Associated with each controller namespace is a namespace ID, labeled as NSID 1 and NSID 2, that is used by the controller to reference a specific namespace. The namespace ID is distinct from the namespace itself and is the handle a host and controller use to specify a particular namespace in a command. The selection of a controller's namespace IDs is outside the scope of this specification. In this example namespace ID 1 is associated

with namespace A and namespace ID 2 is associated with namespace B. Both namespaces are private to the controller and this configuration supports neither multi-path I/O nor namespace sharing.

Figure 16: NVM Express Controller with Two Namespaces

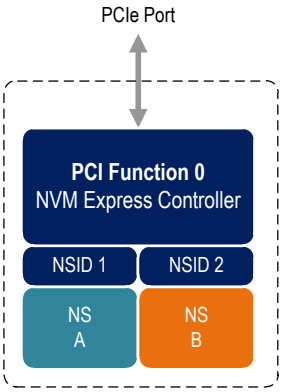
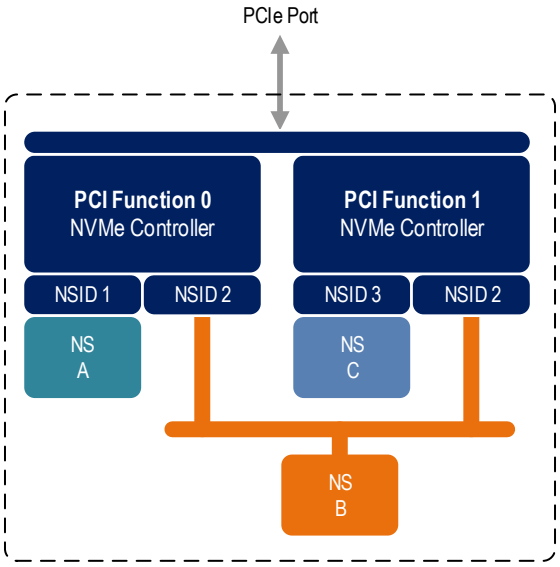


Figure 17 shows a multi-Function NVM subsystem with a single PCI Express port containing two controllers implementing NVMe over PCIe. One controller is associated with PCI Function 0 and the other controller is associated with PCI Function 1. Each controller supports a single private namespace and access to shared namespace B. The namespace ID shall be the same in all controllers that have access to a particular shared namespace. In this example both controllers use namespace ID 2 to access shared namespace B.

Figure 17: NVM Subsystem with Two Controllers and One Port



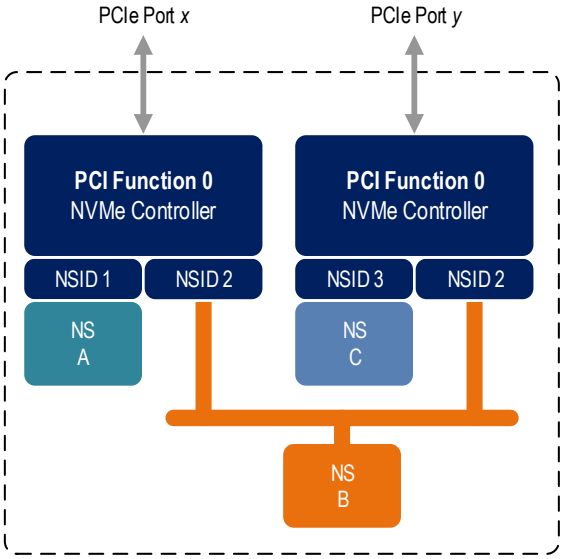
There is one or more Identify Controller data structures for each controller and one or more Identify Namespace data structures for each namespace (refer to Figure 273). Controllers with access to a shared namespace return the Identify Namespace data structure associated with that shared namespace (i.e., the same data structure contents are returned by all controllers with access to the same shared namespace). There is a globally unique identifier associated with the namespace itself and may be used to determine when there are multiple paths to the same shared namespace. Refer to section 4.5.1.

Controllers associated with a shared namespace may operate on the namespace concurrently. Operations performed by individual controllers are atomic to the shared namespace at the write atomicity level of the controller to which the command was submitted (refer to section 3.4.3). The write atomicity level is not required to be the same across controllers that share a namespace. If there are any ordering requirements

between commands issued to different controllers that access a shared namespace, then host software or an associated application, is required to enforce these ordering requirements.

Figure 18 illustrates an NVM subsystem with two PCI Express ports, each with an associated controller implementing NVMe over PCIe. Both controllers map to PCI Function 0 of the corresponding port. Each PCI Express port in this example is completely independent and has its own PCI Express Fundamental Reset and reference clock input. A reset of a port only affects the controller associated with that port and has no impact on the other controller, shared namespace, or operations performed by the other controller on the shared namespace. The functional behavior of this example is otherwise the same as that illustrated in Figure 17.

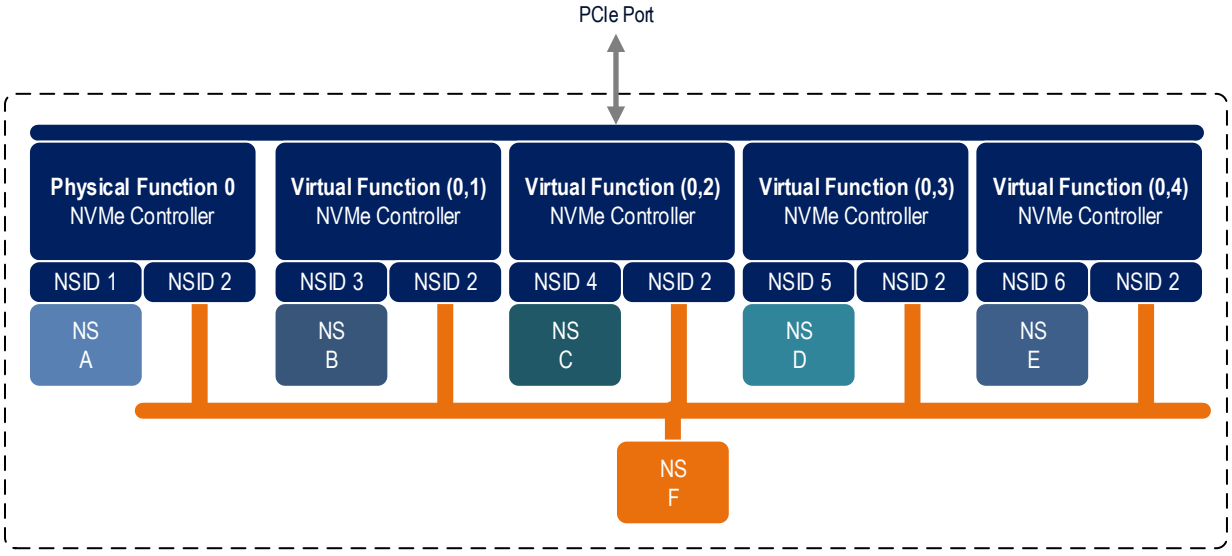
Figure 18: NVM Subsystem with Two Controllers and Two Ports



The two ports shown in Figure 18 may be associated with the same Root Complex or with different Root Complexes and may be used to implement both multi-path I/O and I/O sharing architectures. System-level architectural aspects and use of multiple ports in a PCI Express fabric are beyond the scope of this specification.

Figure 19 illustrates an NVM subsystem that supports Single Root I/O Virtualization (SR-IOV) and has one Physical Function and four Virtual Functions. An NVM Express controller implementing NVMe over PCIe is associated with each Function with each controller having a private namespace and access to a namespace shared by all controllers, labeled NS F. The behavior of the controllers in this example parallels that of the other examples in this section. Refer to section 8.26.4 for more information on SR-IOV.

Figure 19: PCI Express Device Supporting Single Root I/O Virtualization (SR-IOV)



Examples provided in this section are meant to illustrate concepts and are not intended to enumerate all possible configurations. For example, an NVM subsystem may contain multiple PCI Express ports with each port supporting SR-IOV.

2.4.2 Asymmetric Controller Behavior

Asymmetric controller behavior occurs in NVM subsystems where namespace access characteristics (e.g., performance) may vary based on:

- the internal configuration of the NVM subsystem; or
- which controller is used to access a namespace (e.g., Fabrics).

NVM subsystems that provide asymmetric controller behavior may support Asymmetric Namespace Access Reporting as described in section 8.1.

3 NVM Express Architecture

3.1 NVM Controller Architecture

A controller is the interface between a host and an NVM subsystem.

3.1.1 Controller Model

This specification defines two controller models. An NVM subsystem may support a static or dynamic controller model. All controllers in the NVM subsystem shall follow the same controller model.

In a static controller model, controllers that may be allocated to a particular host may have different state at the time the association is established. The controllers within an NVM subsystem are distinguished by their controller identifier. All memory-based transport model controllers shall support the static controller model.

In a dynamic controller model, the controller is allocated by the NVM subsystem on demand. In this model, all controllers allocated to a specific host have the same state at the time the association is established, including attached namespaces and Feature settings. Changes to a controller (e.g., attached namespaces, Feature settings) after the association is established do not impact other dynamic controllers.

Controllers using the message-based transport model in an NVM subsystem may use a dynamic or static controller model. A Discovery controller shall support the dynamic controller model.

An association is established between a host and a controller when the host connects to a controller's Admin Queue using the Fabrics Connect command (refer to section 6.3). Within the Connect command, the host specifies the Host NQN, NVM Subsystem NQN, Host Identifier, and may request a specific Controller ID or may request a connection to any available controller. A controller has only one association at a time.

In a dynamic controller model, the controller is allocated by the NVM subsystem on demand with no state (e.g., Feature settings) preserved from prior associations. In a static controller model, the host may request a particular controller based on the Controller ID where state (e.g., Feature settings) is preserved from prior associations.

While an association exists between a host and a controller, only that host may establish connections with I/O Queues of that controller by presenting the same Host NQN, Host Identifier, NVM Subsystem NQN and Controller ID in subsequent Connect command(s) using the same NVM subsystem port, NVMe Transport type, and NVMe Transport address.

An association between a host and controller is terminated if:

- the controller is shutdown as described in section 3.6.2;
- a Controller Level Reset occurs;
- the NVMe Transport connection is lost between the host and controller for the Admin Queue or any I/O Queue; or
- an NVMe Transport connection is lost between the host and controller for any I/O Queue and the host or controller does not support individual I/O Queue deletion (refer to section 3.3.2.4).

There is no explicit NVMe command that breaks the NVMe Transport association between a host and controller. The Disconnect command (refer to section 6.4) provides a method to delete an NVMe I/O Queue (refer to section 3.3.2.4). While a controller is associated with a host, that controller is busy, and no other associations may be made with that controller.

To use the dynamic controller model, the host specifies a controller identifier of FFFFh when using the Fabrics Connect command (refer to section 6.3) to establish an association with an NVM subsystem.

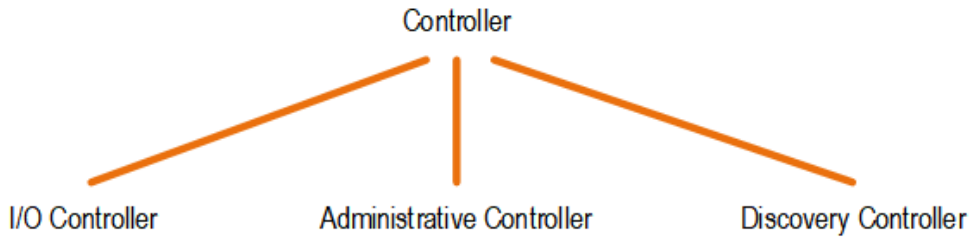
When using the static controller model with a Fabric connected controller, the state that persists across associations is any state that persists across a Controller Level Reset. Additionally, different controllers may present different Feature settings or namespace attachments to the same host. The NVM subsystem may allocate particular controllers to specific hosts.

While allocation of static controllers to hosts are expected to be durable (so that hosts can expect to form associations to the same controllers repeatedly (e.g., after each host reboot)), the NVM subsystem may remove the host allocation of a controller that is not in use at any time for implementation specific reasons (e.g., controller resource reclamation, subsystem reconfiguration).

3.1.2 Controller Types

As shown in Figure 20, there are three types of controllers. An I/O controller (refer to section 3.1.2.1) is a controller that supports commands that provide access to user data stored on an NVM subsystem's non-volatile storage medium and may support commands that provide management capabilities. An Administrative controller (refer to section 3.1.2.2) is a controller that supports commands that provide management capabilities, but does not support I/O commands that access to user data stored on an NVM subsystem's non-volatile storage medium. A Discovery controller (refer to section 3.1.2.3) is a controller used in NVMe over Fabrics to provide access to a Discovery Log Page.

Figure 20: Controller Types



The Controller Type (CNTRLTYPE) field in the Identify Controller data structure indicates a controller's type. Regardless of controller type, all controllers implement one Admin Submission Queue and one Admin Completion Queue. Depending on the controller type, a controller may also support one or more I/O Submission Queues and I/O Completion Queues.

When using a memory-based transport implementation (e.g. PCIe), host software submits commands to a controller through pre-allocated Submission Queues. A controller is alerted to newly submitted commands through SQ Tail Doorbell register writes. The difference between the previous doorbell register value and the current register write indicates the number of commands that were submitted.

A controller fetches commands from the Submission Queue(s) and processes them. Except for fused operations, there are no ordering restrictions for processing of commands within or across Submission Queues. Host software should not submit commands to a Submission Queue that may not be re-ordered arbitrarily. Data associated with the processing of a command may or may not be committed to the NVM subsystem non-volatile memory storage medium in the order that commands are submitted.

Host software submits commands of higher priorities to the appropriate Submission Queues. Priority is associated with the Submission Queue itself, thus the priority of the command is based on the Submission Queue to which that command was submitted. The controller arbitrates across the Submission Queues based on fairness and priority according to the arbitration scheme specified in section 3.4.4.

Upon completion of the command execution by the NVM subsystem, the controller presents completion queue entries to the host through the appropriate Completion Queues. Transport specific methods (e.g., PCIe interrupts) are used to notify the host of completion queue entries to process (refer to the appropriate Transport specification).

There are no ordering restrictions for completions to the host. Each completion queue entry identifies the Submission Queue Identifier and Command Identifier of the associated command. Host software uses this information to correlate the completions with the commands submitted to the Submission Queue(s).

Host software is responsible for creating I/O Submission Queues and I/O Completion Queues prior to using those queue pairs to submit commands to the controller. I/O Submission Queues and I/O Completion Queues are created using the Create I/O Submission Queue command (refer to section 5.5) and the Create I/O Completion Queue command (refer to section 5.4).

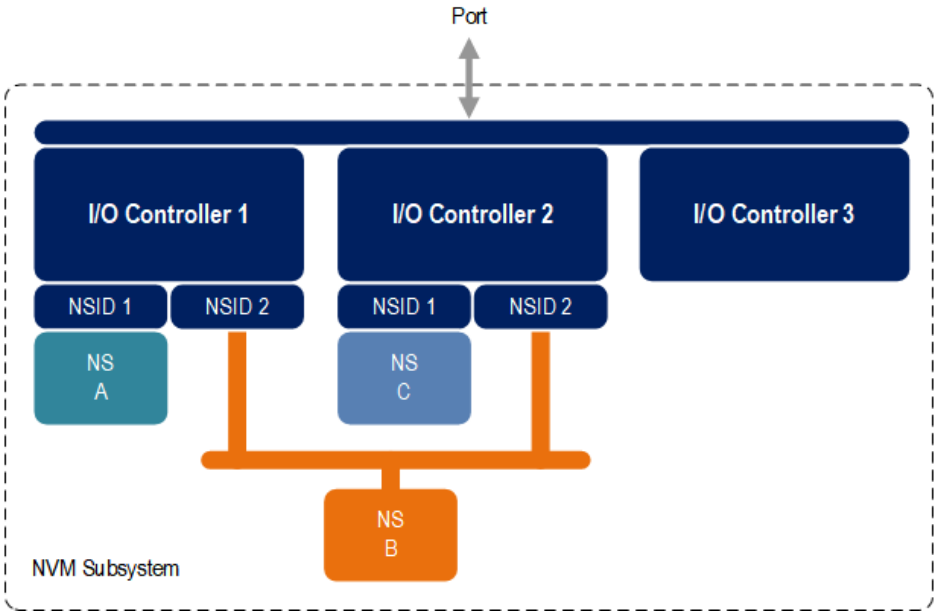
3.1.2.1 I/O Controller

An I/O controller is a controller that supports commands that provide access to user data stored on an NVM subsystem's non-volatile storage medium using an I/O command set and may support commands that provide management capabilities.

An I/O controller may simultaneously support multiple I/O Command Sets. The I/O Command Sets that the controller supports and which of these I/O Command Sets the controller simultaneously supports is reported in the Identify I/O Command Set data structure (refer to section 5.17.2.21). The contents of the Identify I/O Command Set data structure are not required to be the same for all controllers in an NVM subsystem.

Figure 21 shows an NVM subsystem with three I/O controllers. I/O controller one has two attached namespaces, private namespace A and shared namespace B. I/O controller two also has two attached namespaces, private namespace C and shared namespace B. I/O controller three has no attached namespaces. At some later point in time shared namespace B may be attached to I/O controller three.

Figure 21: NVM Subsystem with Three I/O Controllers



3.1.2.1.1 Command Support

Figure 22 and Figure 23 defines commands that are mandatory, optional, and prohibited for an I/O controller. I/O Command Set specific command support requirements are described within individual I/O Command Set specifications.

Figure 22: I/O Controller – Admin Command Support

Command	Command Support Requirements [1]	Reference
Delete I/O Submission Queue	M	5.7
Create I/O Submission Queue	M	5.5
Get Log Page	M	5.16
Delete I/O Completion Queue	M	5.6
Create I/O Completion Queue	M	5.4
Identify	M	5.17
Abort	M	5.1
Set Features	M	5.27
Get Features	M	5.15

Figure 22: I/O Controller – Admin Command Support

Command	Command Support Requirements [1]	Reference
Asynchronous Event Request	M	5.2
Capacity Management	O	5.3
Namespace Management	O	5.23
Firmware Commit	O	5.12
Firmware Image Download	O	5.13
Device Self-test	O	5.9
Namespace Attachment	O	5.22
Keep Alive	NOTE 2	5.18
Directive Send	O	5.11
Directive Receive	O	5.10
Virtualization Management	O	5.28
NVMe-MI Send	O	5.21
NVMe-MI Receive	O	5.20
Doorbell Buffer Config	O	5.8
Lockdown	O	5.19
Format NVM	O	5.14
Security Send	O	5.26
Security Receive	O	5.25
Sanitize	O	5.24
Property Set	M[3]	6.6
Connect	M[3]	6.3
Property Get	M[3]	6.5
Authentication Send	O[3]	6.2
Authentication Receive	O[3]	6.1
Disconnect	O[3]	6.4
I/O Command Set specific Admin Command	Refer to the applicable I/O Command Set specification	Refer to the applicable I/O Command Set specification
Vendor Specific	O	

Notes:
1. O/M/P definition: O = Optional, M = Mandatory, P = Prohibited
2. For NVMe over PCIe implementations, the Keep Alive command is optional. For NVMe over Fabrics implementations, the associated NVMe Transport binding defines whether the Keep Alive command is optional or mandatory.
3. For NVMe over PCIe implementations, all Fabrics commands are prohibited. For NVMe over Fabrics implementations, the commands are as noted in the table.

Figure 23: I/O Controller –Common I/O Command Support

Command	Command Support Requirements [1]
Flush	M
Dataset Management	O
Reservation Register	O [2]
Reservation Report	O [2]
Reservation Acquire	O [2]
Reservation Release	O [2]

Notes:
1. O/M/P definition: O = Optional, M = Mandatory, P = Prohibited
2. Mandatory if reservations are supported as indicated in the Identify Controller data structure.

3.1.2.1.2 Log Page Support

Figure 24 defines log pages that are mandatory, optional, and prohibited for an I/O controller. I/O Command Set specific log page support requirements are described within individual I/O Command Set specifications.

Figure 24: I/O Controller – Log Page Support

Log Page Name	Log Page Support Requirements [1]
Supported Log Pages	M[3]
Error Information	M
SMART / Health Information (Controller scope)	M
SMART / Health Information (Namespace scope)	O
Firmware Slot Information	M
Changed Namespace List	O
Commands Supported and Effects	M[3]
Device Self-test	O
Telemetry Host-Initiated	O
Telemetry Controller-Initiated	O
Endurance Group Information	O
Predictable Latency Per NVM Set	O
Predictable Latency Event Aggregate	O
Asymmetric Namespace Access	O
Persistent Event	O
Endurance Group Event Aggregate	O
Media Unit Status	O[2]
Supported Capacity Configuration List	O[2]
Feature Identifiers Supported and Effects	M[3]
NVMe-MI Commands Supported and Effects	M[3]
Command and Feature Lockdown	O
Reservation Notification	O
Sanitize Status	O
Boot Partition	O
Rotational Media Information	O

Notes:
1. O/M/P definition: O = Optional, M = Mandatory, P = Prohibited
2. Mandatory for controllers that support Fixed Capacity Management (refer to section 8.3.2).
3. Optional for NVM Express revision 1.4 and earlier.

3.1.2.1.3 Features Support

Figure 25 defines features that are mandatory, optional, and prohibited for an I/O controller. I/O Command Set specific feature support requirements are described within individual I/O Command Set specifications.

Figure 25: I/O Controller – Feature Support

Feature Name	Feature Support Requirements [1]	Logged in Persistent Event Log [1]
Arbitration	M	O
Power Management	M	NR
Temperature Threshold	M	O
Volatile Write Cache	O	O
Number of Queues	M	O
Interrupt Coalescing	NOTE 2	O
Interrupt Vector Configuration	NOTE 2	O
Asynchronous Event Configuration	M	NR

Figure 25: I/O Controller – Feature Support

Feature Name	Feature Support Requirements [1]	Logged in Persistent Event Log [1]
Autonomous Power State Transition	O	O
Host Memory Buffer	O	O
Timestamp	O	P
Keep Alive Timer	O	O
Host Controlled Thermal Management	O	O
Non-Operational Power State Config	O	O
Read Recovery Level Config	O	O
Predictable Latency Mode Config	O	O
Predictable Latency Mode Window	O	P
Host Behavior Support	O	O
Sanitize Config	O	O
Endurance Group Event Configuration	O	O
I/O Command Set Profile	O	O
Software Progress Marker	O	NR
Host Identifier	O[3]	O
Reservation Notification Mask	O[4]	O
Reservation Persistence	O[4]	O
Namespace Write Protection Config	O	O
Enhanced Controller Metadata	O[5]	O
Controller Metadata	O[5]	O
Namespace Metadata	O[5]	O
Rotational Media	O	O

Notes:
1. O/M/P/NR definition: O = Optional, M = Mandatory, P = Prohibited, NR = Not Recommended
2. The feature is mandatory for NVMe over PCIe. This feature is not supported for NVMe over Fabrics.
3. Mandatory if reservations are supported as indicated in the Identify Controller data structure.
4. Mandatory if reservations are supported by the namespace as indicated by a non-zero value in the Reservation Capabilities (RESCAP) field in the Identify Namespace data structure.
5. This feature is optional for NVM subsystems that do not implement a Management Endpoint. For NVM subsystems that implement any Management Endpoint refer to the NVM Express Management Interface Specification.

3.1.2.2 Administrative Controller

An Administrative controller is a controller whose intended purpose is to provide NVM subsystem management capabilities. While an I/O controller may support these same management capabilities, an Administrative controller has fewer mandatory capabilities. Unlike an I/O controller, an Administrative controller does not support I/O commands that access to user data stored on an NVM subsystem's non-volatile storage medium. NVMe Transports may support a transport specific mechanism to allow an Administrative controller to load a dedicated NVMe management driver instead of a generic NVMe driver (refer to the applicable NVMe Transport binding specification for details).

Examples of management capabilities that may be supported by an Administrative controller include the following.

- Ability to efficiently poll NVM subsystem health status via NVMe-MI using the NVMe-MI Send command and the NVMe-MI Receive command (refer to the NVM Subsystem Health Status Poll section in the NVM Express Management Interface Specification);
- Ability to manage an NVMe enclosure via NVMe-MI using the NVMe-MI Send command and the NVMe-MI Receive command;
- Ability to manage NVM subsystem namespaces using the Namespace Attachment command and the Namespace Management command;
- Ability to perform virtualization management using the Virtualization Management command;

- Ability to reset an entire NVM subsystem using the NVM Subsystem Reset (NSSR) register, if supported; and
- Ability to shutdown an entire NVM subsystem using the NVM Subsystem Shutdown (NSSD) property, if supported.

An Administrative controller shall not support I/O queues. Namespaces shall not be attached to an Administrative controller.

An Administrative controller is required to support the mandatory Admin commands listed in Figure 28. An Administrative controller may support one or more I/O Command Sets. When an Administrative controller supports an I/O Command Set, then only I/O Command Set specific Admin commands may be supported since an Administrative controller only has an Admin Queue and no I/O Queues.

Figure 26 shows an NVM subsystem with one Administrative controller and two I/O controllers within an NVM subsystem that contains a non-volatile storage medium and namespaces. I/O controller one has two attached namespaces, private namespace A and shared namespace B. I/O controller two also has two attached namespaces, private namespace C and shared namespace B. Since an Administrative controller does not provide access to user data stored on an NVM subsystem's non-volatile storage medium, the Administrative controller has no attached namespaces. The Administrative controller in this example may be used for tasks such as NVM subsystem namespace management and efficiently polling NVM subsystem health status via NVMe-MI. While this example shows a single Administrative controller, an NVM subsystem may support zero or more Administrative controllers.

Figure 26: NVM Subsystem with One Administrative and Two I/O Controllers

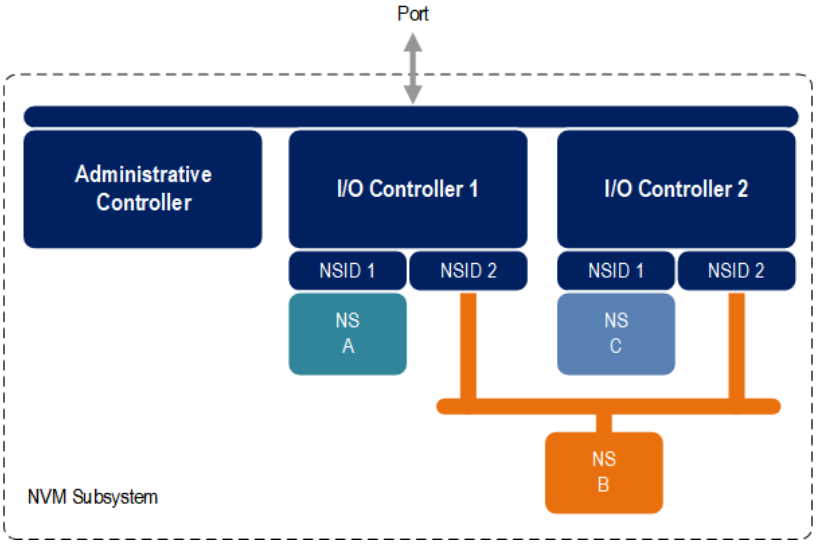
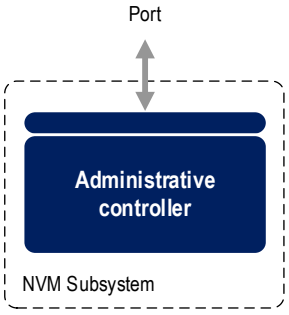


Figure 27 shows an NVM subsystem with one Administrative controller within an NVM subsystem that contains no non-volatile storage medium or namespaces. The Administrative controller in this example may be used to manage an NVMe enclosure using NVMe-MI. Since the Administrative controller is used for a very specific dedicated purpose, the implementer of such an Administrative controller may choose to implement only the mandatory capabilities along with the NVMe-MI Send and NVMe-MI Receive commands.

Figure 27: NVM Subsystem with One Administrative Controller



3.1.2.2.1 Command Support

Figure 28 defines commands that are mandatory, optional, and prohibited for an Administrative controller. Since an Administrative controller does not support I/O queues, NVM Command Set commands that are not admin commands are not supported. A host may utilize the Commands Supported and Effects log page to determine optional commands that are supported by an Administrative controller.

Figure 28: Administrative Controller – Admin Command Support

Command	Command Support Requirements [1]	Reference
Delete I/O Submission Queue	P	5.7
Create I/O Submission Queue	P	5.5
Get Log Page	M	5.16
Delete I/O Completion Queue	P	5.6
Create I/O Completion Queue	P	5.4
Identify	M	5.17
Abort	O	5.1
Set Features	O [3]	5.27
Get Features	O [3]	5.15
Asynchronous Event Request	O [4]	5.2
Capacity Management	O	5.3
Namespace Management	O	5.23
Firmware Commit	O	5.12
Firmware Image Download	O	5.13
Device Self-test	O	5.9
Namespace Attachment	O	5.22
Keep Alive	NOTE 2	5.18
Directive Send	O	5.11
Directive Receive	O	5.10
Virtualization Management	O	5.28
NVMe-MI Send	O	5.21
NVMe-MI Receive	O	5.20
Doorbell Buffer Config	O	5.8
Lockdown	O	5.19
Format NVM	O	5.14
Security Send	O	5.26
Security Receive	O	5.25
Sanitize	O	5.24
Property Set	M [5]	6.6
Connect	M [5]	6.3

Figure 28: Administrative Controller – Admin Command Support

Command	Command Support Requirements [1]	Reference
Property Get	M[5]	6.5
Authentication Send	O[5]	6.2
Authentication Receive	O[5]	6.1
Disconnect	P	6.4
I/O Command Set Specific Admin Commands	P	
Vendor Specific	O	

Notes:
1. O/M/P definition: O = Optional, M = Mandatory, P = Prohibited
2. For NVMe over PCIe implementations, the Keep Alive command is optional. For NVMe over Fabrics implementations, the associated NVMe Transport binding defines whether the Keep Alive command is optional or mandatory.
3. Mandatory if any of the features in Figure 30 are implemented.
4. Mandatory if Telemetry Log, Firmware Commit, or SMART/Health Critical Warnings are supported.
5. For NVMe over PCIe implementations, all Fabrics commands are prohibited. For NVMe over Fabrics implementations, the commands are as noted in the table.

3.1.2.2.2 Log Page Support

Figure 29 defines log pages that are mandatory, optional, and prohibited for an Administrative controller.

Figure 29: Administrative Controller – Log Page Support

Log Page Name	Command Support Requirements [1]
Supported Log Pages	M[4]
Error Information	M
SMART / Health Information (Controller scope)	O
SMART / Health Information (Namespace scope)	O
Firmware Slot Information	O
Changed Namespace List	O
Commands Supported and Effects	M
Device Self-test	O
Telemetry Host-Initiated	O
Endurance Group Information	O
Predictable Latency Per NVM Set	O
Predictable Latency Event Aggregate	O
Asymmetric Namespace Access	P
Persistent Event	O
Endurance Group Event Aggregate	O
Media Unit Status	P
Supported Capacity Configuration List	P
Feature Identifiers Supported and Effects	M[2,4]
NVMe-MI Commands Supported and Effects	M[3,4]
Command and Feature Lockdown	O
Reservation Notification	P
Sanitize Status	O
Rotational Media	P
Boot Partition	O

Figure 29: Administrative Controller – Log Page Support

Log Page Name	Command Support Requirements [1]
I/O Command Set Specific Log Pages	P

Notes:
1. O/M/P definition: O = Optional, M = Mandatory, P = Prohibited
2. Optional if Set Features command is not supported (refer to Figure 28).
3. Optional if NVMe-MI Send command and NVMe-MI Receive command is not supported (refer to Figure 28).
4. Optional for NVM Express revision 1.4 and earlier.

3.1.2.2.3 Features Support

Figure 30 defines features that are mandatory, optional, and prohibited for an Administrative controller. If any feature is supported, then the Set Features and Get Features commands shall be supported. I/O Command Set specific feature support requirements for I/O Controllers are described within individual I/O Command Set specification.

Figure 30: Administrative Controller – Feature Support

Feature Name	Feature Support Requirements [1]	Logged in Persistent Event Log [1]
Arbitration	P	P
Power Management	O	NR
Temperature Threshold	O	O
Volatile Write Cache	P	P
Number of Queues	P	P
Interrupt Coalescing	NOTE 2	NOTE 2
Interrupt Vector Configuration	NOTE 2	NOTE 2
Asynchronous Event Configuration	O[3]	NR
Autonomous Power State Transition	O	O
Host Memory Buffer	O	O
Timestamp	O	P
Keep Alive Timer	O	O
Host Controlled Thermal Management	O	O
Non-Operational Power State Config	O	O
Read Recovery Level Config	O	O
Predictable Latency Mode Config	O	P
Predictable Latency Mode Window	O	O
Host Behavior Support	O	O
Sanitize Config	O	O
Endurance Group Event Configuration	O	O
I/O Command Set Profile	P	P
Software Progress Marker	O	NR
Host Identifier	O[4]	O
Reservation Notification Mask	O[5]	O
Reservation Persistence	O[5]	O
Namespace Write Protection Config	O	O
Enhanced Controller Metadata	O[6]	O
Controller Metadata	O[6]	O
Namespace Metadata	O[6]	O
Rotational Media	P	P

Figure 30: Administrative Controller – Feature Support

Feature Name	Feature Support Requirements [1]	Logged in Persistent Event Log [1]
Notes: 1. O/M/P/NR definition: O = Optional, M = Mandatory, P = Prohibited, NR = Not Recommended. 2. The feature is mandatory for NVMe over PCIe. This feature is not supported for NVMe over Fabrics. 3. Mandatory if Telemetry Log, Firmware Commit or SMART/Health Critical Warnings are supported. 4. Mandatory if reservations are supported as indicated in the Identify Controller data structure. 5. Mandatory if reservations are supported by the namespace as indicated by a non-zero value in the Reservation Capabilities (RESCAP) field in the Identify Namespace data structure. 6. This feature is optional for NVM subsystems that do not implement a Management Endpoint. For NVM subsystems that implement any Management Endpoint refer to the NVM Express Management Interface Specification.		

3.1.2.3 Discovery Controller

A Discovery controller only implements features related to Discovery Log Pages and does not implement I/O Queues, I/O commands, or expose namespaces. The functionality supported by the Discovery controller is defined in section 3.1.2.3.4.

The host uses the well-known Discovery Service NQN (nqn.2014-08.org.nvmexpress.discovery) in the Connect command (refer to section 6.3) to a Discovery Service. The method that a host uses to obtain the NVMe Transport information necessary to connect to the well-known Discovery Service is implementation specific.

The Discovery Log Page provided by a Discovery controller contains one or more entries. Each entry specifies information necessary for the host to connect to an NVM subsystem. An entry may be associated with an NVM subsystem that exposes namespaces or a referral to another Discovery Service. There are no ordering requirements for log page entries within the Discovery Log Page.

Discovery controller(s) may provide different log page contents depending on the Host NQN provided (e.g., different NVM subsystems may be accessible to different hosts). The set of Discovery Log Page Entries should include all applicable addresses on the same fabric as the Discovery Service and may include addresses on other fabrics.

Discovery controllers that support explicit persistent connections shall support both Asynchronous Event Request and Keep Alive commands (refer to sections 5.2 and 5.18 respectively). A host requests an explicit persistent connection to a Discovery controller and Asynchronous Event Notifications from the Discovery controller on that persistent connection by specifying a non-zero Keep Alive Timer value in the Connect command. If the Connect command specifies a non-zero Keep Alive Timer value and the Discovery controller does not support Asynchronous Events, then the Discovery controller shall return a status value of Connect Invalid Parameters (refer to Figure 383) for the Connect command. Discovery controllers shall indicate support for Discovery Log Change Notifications in the Identify Controller Data Structure (refer to Figure 275).

Discovery controllers that do not support explicit persistent connections shall not support Keep Alive commands and may use a fixed Discovery controller activity timeout value (e.g., 2 minutes). If no commands are received by such a Discovery controller within that time period, the controller may perform the actions for Keep Alive Timer expiration defined in section 3.9.

A Discovery controller shall not support the Disconnect command.

A Discovery Log Page with multiple Discovery Log Page Entries for the same NVM subsystem indicates that there are multiple fabric paths to the NVM subsystem, and/or that multiple static controllers may share a fabric path. The host may use this information to form multiple associations to controllers within an NVM subsystem.

Multiple Discovery Log Page Entries for the same NVM subsystem with different Port ID values indicates that the resulting NVMe Transport connections are independent with respect to NVM subsystem port

hardware failures. A host that uses a single association should pick a record to attach to an NVM subsystem. A host that uses multiple associations should choose different ports.

A transport specific method may exist to indicate changes to a Discovery controller.

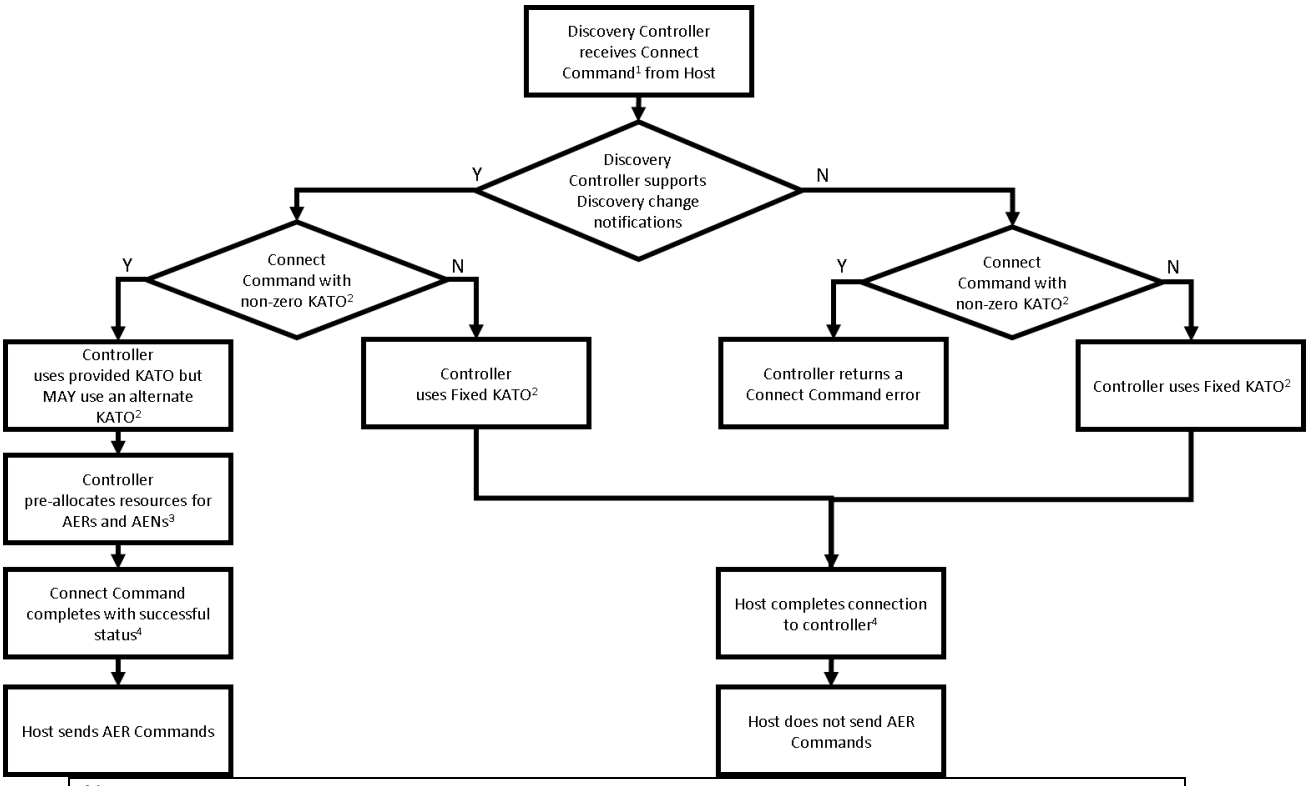
The Controller ID values returned in the Discovery Log Page Entries indicate whether an NVM subsystem supports the dynamic or static controller model. The controller ID value of FFFFh is a special value used for NVM subsystems that support the dynamic controller model indicating that any available controller may be returned. The Controller ID value of FFFEh is a special value used for NVM subsystems that support the static controller model indicating that any available controller may be returned. An NVM subsystem supports the dynamic controller model if Discovery Log Page Entries use the Controller ID value of FFFFh. An NVM subsystem supports the static controller model if Discovery Log Page Entries use a Controller ID value that is less than FFFFh. The Identify Controller data structure also indicates whether an NVM subsystem is dynamic or static.

If an NVM subsystem implements the dynamic controller model, then multiple Discovery Log Page Entries (refer to Figure 264) with the Controller ID set to FFFFh may be returned for that NVM subsystem (e.g., to indicate multiple NVM subsystem ports) in the Discovery Log Page. If an NVM subsystem implements the static controller model, then multiple Discovery Log Page Entries that indicate different Controller ID values may be returned for that NVM subsystem in the Discovery Log Page. If an NVM subsystem that implements the static controller model includes any Discovery Log Page Entries that indicate a Controller ID of FFFEh, then the host should remember the Controller ID returned from the Fabrics Connect command and re-use the allocated Controller ID for future associations to that particular controller.

3.1.2.3.1 Discovery Controller Initialization

The initialization process for Discovery controllers is described in Figure 31.

Figure 31: Discovery Controller Initialization process flow



Notes:

1. Refer to section 6.3;
2. Refer to the Asynchronous Event Request command in section 5.2;
3. Refer to the Keep Alive command in section 5.18; and
4. Refer to the following steps in this section.

After the Connect Command completes with a status of Successful Completion, the host performs the following steps:

1. NVMe authentication is performed if required (refer to section 8.13.2);
2. The host determines the controller's capabilities by reading the Controller Capabilities property;
3. The host configures the controller's settings by writing the Controller Configuration property, including setting CC.EN to '1' to enable command processing;
4. The host waits for the controller to indicate that the controller is ready to process commands. The controller is ready to process commands when CSTS.RDY is set to '1' in the Controller Status property; and
5. The host determines the features and capabilities of the controller by issuing an Identify command, specifying each applicable Controller data structure.

After initializing the Discovery controller, the host reads the Discovery Log Page. Refer to section 5.16.1.21.

3.1.2.3.2 Command Support

A Discovery controller supports all mandatory Fabrics commands. A Discovery controller supports a subset of Admin commands shown in Figure 32.

Figure 32: Discovery Controller – Admin Command Support

Command	Command Support Requirements [1]	Reference
Delete I/O Submission Queue	P	5.7
Create I/O Submission Queue	P	5.5

Figure 32: Discovery Controller – Admin Command Support

Command	Command Support Requirements [1]	Reference
Get Log Page	M	5.16
Delete I/O Completion Queue	P	5.6
Create I/O Completion Queue	P	5.4
Identify	M	5.17
Abort	O	5.1
Set Features	NOTE 2	5.27
Get Features	NOTE 2	5.15
Asynchronous Event Request	NOTE 2	5.2
Capacity Management	P	5.3
Namespace Management	P	5.23
Firmware Commit	P	5.12
Firmware Image Download	P	5.13
Device Self-test	P	5.9
Namespace Attachment	P	5.22
Keep Alive	NOTE 2	5.18
Directive Send	P	5.11
Directive Receive	P	5.10
Virtualization Management	P	5.28
NVMe-MI Send	P	5.21
NVMe-MI Receive	P	5.20
Doorbell Buffer Config	P	5.8
Lockdown	P	5.19
Format NVM	P	5.14
Security Send	P	5.26
Security Receive	P	5.25
Sanitize	P	5.24
Property Set	M	6.6
Connect	M	6.3
Property Get	M	6.5
Authentication Send	O	6.2
Authentication Receive	O	6.1
Disconnect	P	6.4
I/O Command Set Specific Admin Commands	P	
Vendor Specific	O	

Notes:
1. O/M/P definition: O = Optional, M = Mandatory, P = Prohibited
2. For Discovery controllers that do not support explicit persistent connections, the command is prohibited. For Discovery controllers that support explicit persistent connections, the command is mandatory.

3.1.2.3.3 Log Page Support

The Discovery controller shall support the Discovery Log Page. The log pages that a Discovery controller may support are shown in Figure 33.

Figure 33: Discovery Controller – Log Page Support

Log Page Name	Command Support Requirements [1]
Supported Log Pages	M[4]
Error Information	O
SMART / Health Information (Controller scope)	P
SMART / Health Information (Namespace scope)	P
Firmware Slot Information	P

Figure 33: Discovery Controller – Log Page Support

Log Page Name	Command Support Requirements [1]
Changed Namespace List	P
Commands Supported and Effects	P
Device Self-test	P
Telemetry Host-Initiated	P
Telemetry Controller-Initiated	P
Endurance Group Information	P
Predictable Latency Per NVM Set	P
Predictable Latency Event Aggregate	P
Asymmetric Namespace Access	P
Persistent Event	P
Endurance Group Event Aggregate	P
Media Unit Status	P
Supported Capacity Configuration List	P
Feature Identifiers Supported and Effects	M[2,4]
NVMe-MI Commands Supported and Effects	M[3,4]
Command and Feature Lockdown	P
Discovery	M
Reservation Notification	P
Sanitize Status	P
Rotational Media	P
Boot Partition	P
I/O Command Set Specific Log Pages	P

Notes
1. O/M/P definition: O = Optional, M = Mandatory, P = Prohibited
2. Optional if Set Features command is not supported (refer to Figure 32).
3. Optional if NVMe-MI Send command and NVMe-MI Receive command is not supported (refer to Figure 32).
4. Optional for versions 1.1 and earlier of the NVMe over Fabrics specification.

3.1.2.3.4 Features Support

These features indicate the attributes of a Discovery controller (refer to Figure 34). This is optional information not required for proper behavior of the system (refer to Figure 316).

Figure 34: Discovery Controller – Feature Support

Feature Name	Feature Support Requirements [1]	Logged in Persistent Event Log [1]
Arbitration	P	P
Power Management	P	P
Temperature Threshold	P	P
Volatile Write Cache	P	P
Number of Queues	P	P
Interrupt Coalescing	P	P
Interrupt Vector Configuration	P	P
Asynchronous Event Configuration	O	NR
Autonomous Power State Transition	P	P
Host Memory Buffer	P	P
Timestamp	P	P
Keep Alive Timer	O	O
Host Controlled Thermal Management	P	P
Non-Operational Power State Config	P	P
Read Recovery Level Config	P	P

Figure 34: Discovery Controller – Feature Support

Feature Name	Feature Support Requirements [1]	Logged in Persistent Event Log [1]
Predictable Latency Mode Config	P	P
Predictable Latency Mode Window	P	P
Host Behavior Support	P	P
Sanitize Config	P	P
Endurance Group Event Configuration	P	P
Vendor Specific	O	O
Software Progress Marker	P	P
Host Identifier	P	P
Reservation Notification Mask	P	P
Reservation Persistence	P	P
Namespace Write Protection Config	P	P
I/O Command Set Profile	P	P
Enhanced Controller Metadata	O[2]	O
Controller Metadata	O[2]	O
Namespace Metadata	O[2]	O
Rotational Media	P	P

Notes:
1. O/M/P/NR definition: O = Optional, M = Mandatory, P = Prohibited, NR = Not Recommended.
2. This feature is optional for NVM subsystems that do not implement a Management Endpoint. For NVM subsystems that implement any Management Endpoint refer to the NVM Express Management Interface Specification.

3.1.2.3.4.1 Asynchronous Event Configuration (Feature Identifier 0Bh)

Discovery controllers that support Asynchronous Event Notifications shall implement the Get Features and Set Features commands. A Discovery controller shall enable Asynchronous Discovery Log Event Notifications, if a non-zero Keep Alive Timeout (KATO) value is received in the Connect command (refer to section 6.3) sent to that controller.

Figure 326 defines Discovery controller Asynchronous Event Notifications.

3.1.2.3.4.2 Discovery Controller Asynchronous Event Information – Requests and Notifications

If Discovery controllers detect events about which a host has requested notification, then the Discovery controller shall send an Asynchronous Event with the:

- Asynchronous Event Type field set to Notice (i.e., 2h);
- Log Page Identifier field set to Discovery (i.e., 70h); and
- Asynchronous Event Information field set as defined in Figure 147.

When a Discovery controller updates Discovery Log Page(s), the Discovery controller shall send a Discovery Log Page Change Asynchronous Event notification (Asynchronous Event Information F0h) to each host that has requested asynchronous event notifications of this type (refer to Figure 147).

3.1.3 Controller Properties

A property is a dword, or qword attribute of a controller. The attribute may have read, write, or read/write access. The host shall access a property using the width specified for that property with an offset that is at the beginning of the property unless otherwise noted in a transport specific specification. All reserved properties and all reserved bits within properties are read-only and return 0h when read. Properties may be read with the Property Get command and may be written with the Property Set command with controllers using the message-based transport model. For controllers using the memory-based transport model, refer to the applicable NVMe Transport binding specification for access methods and rules (e.g., NVMe PCIe Transport Specification).

Figure 35 describes the property map for the controller.

Accesses that target any portion of two or more properties are not supported.

Software should not rely on 0h being returned.

Figure 35: Property Definition

Offset (OFST)	Size (in bytes)	I/O Controller[1]	Admin. Controller[1]	Discovery Controller[1]	Name
0h	8	M	M	M	**CAP:** Controller Capabilities
8h	4	M	M	M	**VS:** Version
Ch	4	M[2]	M[2]	R	**INTMS:** Interrupt Mask Set
Fh	4	M[2]	M[2]	R	**INTMC:** Interrupt Mask Clear
14h	4	M	M	M	**CC:** Controller Configuration
18h		R	R	R	Reserved
1Ch	4	M	M	M	**CSTS:** Controller Status
20h	4	O	O	R	**NSSR:** NVM Subsystem Reset
24h	4	M[2]	M[2]	R	**AQA:** Admin Queue Attributes
28h	8	M[2]	M[2]	R	**ASQ:** Admin Submission Queue Base Address
30h	8	M[2]	M[2]	R	**ACQ:** Admin Completion Queue Base Address
38h	4	O[3]	O[3]	R	**CMBLOC:** Controller Memory Buffer Location
3Ch	4	O[3]	O[3]	R	**CMBSZ:** Controller Memory Buffer Size
40h	4	O[3]	O[3]	R	**BPINFO:** Boot Partition Information
44h	4	O[3]	O[3]	R	**BPRSEL:** Boot Partition Read Select
48h	8	O[3]	O[3]	R	**BPMBL:** Boot Partition Memory Buffer Location
50h	8	O[3]	O[3]	R	**CMBMSC:** Controller Memory Buffer Memory Space Control
58h	4	O[3]	O[3]	R	**CMBSTS:** Controller Memory Buffer Status
5Ch	4	O[3]	O[3]	R	**CMBEBS:** Controller Memory Buffer Elasticity Buffer Size
60h	4	O[3]	O[3]	R	**CMBSWTP:** Controller Memory Buffer Sustained Write Throughput
64h	4	O	O	R	**NSSD:** NVM Subsystem Shutdown
68h	4	M	M	R	**CRTO:** Controller Ready Timeouts
6Ch		R	R	R	Reserved
E00h	4	O[3]	O[3]	R	**PMRCAP:** Persistent Memory Capabilities
E04h	4	O[3]	O[3]	R	**PMRCTL:** Persistent Memory Region Control
E08h	4	O[3]	O[3]	R	**PMRSTS:** Persistent Memory Region Status
E0Ch	4	O[3]	O[3]	R	**PMREBS:** Persistent Memory Region Elasticity Buffer Size
E10h	4	O[3]	O[3]	R	**PMRSWTP:** Persistent Memory Region Sustained Write Throughput
E14h	4	O[3]	O[3]	R	**PMRMSCL:** Persistent Memory Region Controller Memory Space Control Lower
E18h	4	O[3]	O[3]	R	**PMRMSCU:** Persistent Memory Region Controller Memory Space Control Upper
E1Ch		R	M	R	Reserved
1000h		T	T	T	Transport Specific
1300h		O	O	O	Vendor Specific
Notes:					

Figure 35: Property Definition

Offset (OFST)	Size (in bytes)	I/O Controller[1]	Admin. Controller[1]	Discovery Controller[1]	Name

1. O/M/P definition: O = Optional, M = Mandatory, R = Reserved, T = Transport Specific
2. Mandatory for memory-based transport implementations. Reserved for message-based transport implementations.
3. Optional for memory-based transport implementations. Reserved for message-based transport implementations.

The following conventions are used to describe controller properties for all transport models. Hardware shall return '0' for all bits that are marked as reserved, and host software shall write all reserved bits and properties with the value of 0h

The following terms and abbreviations are used:

RO	Read Only
RW	Read Write
RWC	Read/Write '1' to clear
RWS	Read/Write '1' to set
Impl Spec	Implementation Specific – the controller has the freedom to choose its implementation.
HwInit	The default state is dependent on NVM Express controller and system configuration.
Reset	This column indicates the value of the field after a Controller Level Reset as defined in section 3.7.2.

For some fields, it is implementation specific as to whether the field is RW, RWC, or RO; this is typically shown as RW/RO or RWC/RO to indicate that if the functionality is not supported that the field is read only.

When a field is referred to in the document, the convention used is "Property Symbol.Field Symbol". For example, the PCI command register Parity Error Response Enable bit is referred to by the name CMD.PEE. If the 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". For example, when the Controller Ready With Media Support sub-field of the Controller Ready Modes Supported field within the Controller Capability property, the sub-field is referred to by the name CAP.CRMS.CRWMS.

3.1.3.1 Offset 0h: CAP – Controller Capabilities

This property indicates basic capabilities of the controller to host software.

Figure 36: Offset 0h: CAP – Controller Capabilities

Bits	Type	Reset	Description
63: 61	RO	0h	Reserved

Figure 36: Offset 0h: CAP – Controller Capabilities

<table>
<tr><th>Bits</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>60:59</td><td>RO</td><td>Impl Spec</td><td>Controller Ready Modes Supported (CRMS): This field indicates the ready capabilities of the controller. Refer to sections 3.5.3 and 3.5.4 for more detail.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>0</td><td>Controller Ready With Media Support (CRWMS): If this bit is set to ‘1’, then the controller supports the Controller Ready With Media mode.

If this bit is cleared to ‘0’, then the controller does not support Controller Ready With Media mode.

This bit shall be set to ‘1’ on controllers compliant with NVM Express Base Specification, Revision 2.0 and later.</td></tr>
<tr><td>1</td><td>Controller Ready Independent of Media Support (CRIMS): If this bit is set to ‘1’, then the controller supports the Controller Ready Independent of Media mode.

If this bit is cleared to ‘0’, then the controller does not support Controller Ready Independent of Media mode.</td></tr>
</table></td></tr>
<tr><td>58</td><td>RO</td><td>Impl Spec</td><td>NVM Subsystem Shutdown Supported (NSSS): This bit indicates whether the controller supports the NVM Subsystem Shutdown feature defined in section 3.6.3 If the controller supports the NVM Subsystem Shutdown feature, then this bit is set to ‘1’. If the controller does not support the NVM Subsystem Shutdown feature, then this bit is cleared to ‘0’. If the NSSRS bit is cleared to ‘0’, then this bit shall be cleared to ‘0’.</td></tr>
<tr><td>57</td><td>RO</td><td>Impl Spec</td><td>Controller Memory Buffer Supported (CMBS): If set to ‘1’, this bit indicates that the controller supports the Controller Memory Buffer, and that addresses supplied by the host are permitted to reference the Controller Memory Buffer only if the host has enabled the Controller Memory Buffer’s controller memory space. If the controller supports the Controller Memory Buffer, this bit shall be set to ‘1’.</td></tr>
<tr><td>56</td><td>RO</td><td>Impl Spec</td><td>Persistent Memory Region Supported (PMRS): This bit indicates whether the Persistent Memory Region is supported. This bit is set to ‘1’ if the Persistent Memory Region is supported. This bit is cleared to ‘0’ if the Persistent Memory Region is not supported.</td></tr>
<tr><td>55:52</td><td>RO</td><td>Impl Spec</td><td>Memory Page Size Maximum (MPSMAX): This field indicates the maximum host memory page size that the controller supports. The maximum memory page size is (2 ^ (12 + MPSMAX)). The host shall not configure a memory page size in CC.MPS that is larger than this value.

For Discovery controllers this field shall be cleared to 0h.</td></tr>
<tr><td>51:48</td><td>RO</td><td>Impl Spec</td><td>Memory Page Size Minimum (MPSMIN): This field indicates the minimum host memory page size that the controller supports. The minimum memory page size is (2 ^ (12 + MPSMIN)). The host shall not configure a memory page size in CC.MPS that is smaller than this value.

For Discovery controllers this shall be cleared to 0h.</td></tr>
<tr><td>47:46</td><td>RO</td><td>Impl Spec</td><td>Controller Power Scope (CPS): This field indicates scope of controlling the main power for this controller.
<table>
<tr><th>Value</th><th>Power Scope</th></tr>
<tr><td>00b</td><td>Not Reported</td></tr>
<tr><td>01b</td><td>Controller scope</td></tr>
<tr><td>10b</td><td>Domain scope (i.e., the NVM subsystem supports multiple domains (refer to section 3.2.4).</td></tr>
<tr><td>11b</td><td>NVM subsystem scope (i.e., the NVM subsystem does not support multiple domains).</td></tr>
</table>
If the NSSS bit is set to ‘1’, then this field shall not be cleared to 00b.</td></tr>
</table>

Figure 36: Offset 0h: CAP – Controller Capabilities

<table>
<tr><th>Bits</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>45</td><td>RO</td><td>Impl Spec</td><td>Boot Partition Support (BPS): This bit indicates whether the controller supports Boot Partitions. If this bit is set to '1', the controller supports Boot Partitions. If this bit is cleared to '0', the controller does not support Boot Partitions. Refer to section 8.2.</td></tr>
<tr><td>44:37</td><td>RO</td><td>Impl Spec</td><td>Command Sets Supported (CSS): This field indicates the I/O Command Set(s) that the controller supports.

For Discovery controllers, this field should have only Bit 0 set to 1.

<table>
<tr><th>Bit</th><th>Definition</th></tr>
<tr><td>7</td><td>No I/O Command Set is supported (i.e., only the Admin Command Set is supported).

This bit shall be set to '1' if no I/O Command Set is supported.</td></tr>
<tr><td>6</td><td>Controller supports one or more I/O Command Sets and supports the Identify I/O Command Set data structure (refer to section 5.17.2.21). Controllers that support I/O Command Sets other than the NVM Command Set shall set this bit to '1'. Controllers that only support the NVM Command Set may set this bit to '1'.</td></tr>
<tr><td>5:1</td><td>Reserved</td></tr>
<tr><td>0</td><td>NVM Command Set or a Discovery controller</td></tr>
</table></td></tr>
<tr><td>36</td><td>RO</td><td>Impl Spec</td><td>NVM Subsystem Reset Supported (NSSRS): This bit indicates whether the controller supports the NVM Subsystem Reset feature defined in section 3.7.1. This bit is set to '1' if the controller supports the NVM Subsystem Reset feature. This bit is cleared to '0' if the controller does not support the NVM Subsystem Reset feature.

For Discovery controllers, this field shall be cleared to 0h.</td></tr>
<tr><td>35:32</td><td>RO</td><td>Impl Spec</td><td>Doorbell Stride (DSTRD): Each Submission Queue and Completion Queue Doorbell property is 32-bits in size. This property indicates the stride between doorbell properties. The stride is specified as (2 ^ (2 + DSTRD)) in bytes. A value of 0h indicates a stride of 4 bytes, where the doorbell properties are packed without reserved space between each property. Refer to section 8.8.

For NVMe over Fabrics I/O controllers, this property shall be cleared to a fixed value of 0h.</td></tr>
</table>

Figure 36: Offset 0h: CAP – Controller Capabilities

Bits	Type	Reset	Description
31:24	RO	Impl Spec	**Timeout (TO):** This is the worst-case time that host software should wait for CSTS.RDY to transition from: a) '0' to '1' after CC.EN transitions from '0' to '1'; or b) '1' to '0' after CC.EN transitions from '1' to '0'. This worst-case time may be experienced after events such as an abrupt shutdown or activation of a new firmware image; typical times are expected to be much shorter. This field is in 500 millisecond units. The maximum value of this field is FFh, which indicates a 127.5 second timeout. If the Controller Ready Independent of Media Enable (CC.CRIME) bit is cleared to '0' and the worst-case time for CSTS.RDY to change state is due to enabling the controller after CC.EN transitions from '0' to '1', then this field shall be set to: a) the value in Controller Ready With Media Timeout (CRTO.CRWMT); or b) FFh if CRTO.CRWMT is greater than FFh. If the Controller Ready Independent of Media Enable (CC.CRIME) bit is set to '1' and the worst-case time for CSTS.RDY to change state is due to enabling the controller after CC.EN transitions from '0' to '1', then this field shall be set to: a) the value in Controller Ready Independent of Media Timeout (CRTO.CRIMT); or b) FFh if CRTO.CRIMT is greater than FFh. Controllers that support the CRTO register (refer to Figure 62) are able to indicate larger timeouts for enabling the controller. Host software should use the value in CRTO.CRWMT or CRTO.CRIMT depending on the controller ready mode indicated by CC.CRIME to determine the worst-case timeout for CSTS.RDY to transition from '0' to '1' after CC.EN transitions from '0' to '1'. Host software that is based on revisions earlier than NVM Express Base Specification, Revision 2.0 is not required to wait for more than 127.5 seconds for CSTS.RDY to transition. Refer to sections 3.5.3 and 3.5.4 for more information.
23:19	RO	0h	Reserved
18:17	RO	Impl Spec	**Arbitration Mechanism Supported (AMS):** This field is bit significant and indicates the optional arbitration mechanisms supported by the controller. If a bit is set to '1', then the corresponding arbitration mechanism is supported by the controller. Refer to section 3.4.4 for arbitration details. **Bit** / **Definition** 0 – Weighted Round Robin with Urgent Priority Class 1 – Vendor Specific The round robin arbitration mechanism is not listed since all controllers shall support this arbitration mechanism. For Discovery controllers, this property shall be cleared to 0h.
16	RO	Impl Spec	**Contiguous Queues Required (CQR):** This bit is set to '1' if the controller requires that I/O Submission Queues and I/O Completion Queues are required to be physically contiguous. This bit is cleared to '0' if the controller supports I/O Submission Queues and I/O Completion Queues that are not physically contiguous. If this bit is set to '1', then the Physically Contiguous bit (CDW11.PC) in the Create I/O Submission Queue and Create I/O Completion Queue commands shall be set to '1'. For I/O controllers and Discovery controllers using a message-based transport, this property shall be set to a value of 1h.

Figure 36: Offset 0h: CAP – Controller Capabilities

Bits	Type	Reset	Description
15:00	RO	Impl Spec	**Maximum Queue Entries Supported (MQES):** This field indicates the maximum individual queue size that the controller supports. For NVMe over PCIe implementations, this value applies to the I/O Submission Queues and I/O Completion Queues that the host creates. For NVMe over Fabrics implementations, this value applies to only the I/O Submission Queues that the host creates. This is a 0's based value. The minimum value is 1h, indicating two entries.

3.1.3.2 Offset 8h: VS – Version

This property indicates the major, minor, and tertiary version of the NVM Express Base Specification that the controller implementation supports. Valid versions of the specification are: 1.0, 1.1, 1.2, 1.2.1, 1.3, 1.4, and 2.0.

3.1.3.2.1 VS Value for 1.0 Compliant Controllers

Figure 37: VS Value for 1.0 Compliant Controllers

Bits	Type	Reset	Description
31:16	RO	1h	**Major Version Number (MJR):** Indicates the major version is "1".
15:08	RO	0h	**Minor Version Number (MNR):** Indicates the minor version is "0".
07:00	RO	0h	Reserved

3.1.3.2.2 VS Value for 1.1 Compliant Controllers

Figure 38: VS Value for 1.1 Compliant Controllers

Bits	Type	Reset	Description
31:16	RO	1h	**Major Version Number (MJR):** Indicates the major version is "1"
15:08	RO	1h	**Minor Version Number (MNR):** Indicates the minor version is "1".
07:00	RO	0h	Reserved

3.1.3.2.3 VS Value for 1.2 Compliant Controllers

Figure 39: VS Value for 1.2 Compliant Controllers

Bits	Type	Reset	Description
31:16	RO	1h	**Major Version Number (MJR):** Indicates the major version is "1".
15:08	RO	2h	**Minor Version Number (MNR):** Indicates the minor version is "2".
07:00	RO	0h	Reserved

3.1.3.2.4 VS Value for 1.2.1 Compliant Controllers

Figure 40: VS Value for 1.2.1 Compliant Controllers

Bits	Type	Reset	Description
31:16	RO	1h	**Major Version Number (MJR):** Indicates the major version is "1".
15:08	RO	2h	**Minor Version Number (MNR):** Indicates the minor version is "2".
07:00	RO	1h	**Tertiary Version Number (TER):** Indicates the tertiary version is "1".

3.1.3.2.5 VS Value for 1.3 Compliant Controllers

Figure 41: VS Value for 1.3 Compliant Controllers

Bits	Type	Reset	Description
31:16	RO	1h	**Major Version Number (MJR):** Indicates the major version is “1”.
15:08	RO	3h	**Minor Version Number (MNR):** Indicates the minor version is “3”.
07:00	RO	0h	**Tertiary Version Number (TER):** Indicates the tertiary version is “0”.

3.1.3.2.6 VS Value for 1.4 Compliant Controllers

Figure 42: VS Value for 1.4 Compliant Controllers

Bits	Type	Reset	Description
31:16	RO	1h	**Major Version Number (MJR):** Indicates the major version is “1”.
15:08	RO	4h	**Minor Version Number (MNR):** Indicates the minor version is “4”.
07:00	RO	0h	**Tertiary Version Number (TER):** Indicates the tertiary version is “0”.

3.1.3.2.7 VS Value for 2.0 Compliant Controllers

Figure 43: VS Value for 2.0 Compliant Controllers

Bits	Type	Reset	Description
31:16	RO	2h	**Major Version Number (MJR):** Indicates the major version is “2”.
15:08	RO	0h	**Minor Version Number (MNR):** Indicates the minor version is “0”.
07:00	RO	0h	**Tertiary Version Number (TER):** Indicates the tertiary version is “0”.

3.1.3.3 Offset Ch: INTMS – Interrupt Mask Set

This property is used to mask interrupts when using pin-based interrupts, single message MSI, or multiple message MSI. When using MSI-X, the interrupt mask table defined as part of MSI-X should be used to mask interrupts. Host software shall not access this property when configured for MSI-X; any accesses when configured for MSI-X is undefined. For interrupt behavior requirements, refer to the Interrupts section of the NVMe over PCIe Transport Specification.

Figure 44: Offset Ch: INTMS – Interrupt Mask Set

Bits	Type	Reset	Description
31:00	RWS	0h	**Interrupt Vector Mask Set (IVMS):** This field is bit significant. If a ‘1’ is written to a bit, then the corresponding interrupt vector is masked from generating an interrupt or reporting a pending interrupt in the MSI Capability Structure. Writing a ‘0’ to a bit has no effect. When read, this field returns the current interrupt mask value within the controller (not the value of this property). If a bit has a value of a ‘1’, then the corresponding interrupt vector is masked. If a bit has a value of ‘0’, then the corresponding interrupt vector is not masked.

3.1.3.4 Offset 10h: INTMC – Interrupt Mask Clear

This property is used to unmask interrupts when using pin-based interrupts, single message MSI, or multiple message MSI. When using MSI-X, the interrupt mask table defined as part of MSI-X should be used to unmask interrupts. Host software shall not access this property when configured for MSI-X; any accesses

when configured for MSI-X is undefined. For interrupt behavior requirements, refer to the Interrupts section of the NVMe over PCIe Transport Specification.

Figure 45: Offset 10h: INTMC – Interrupt Mask Clear

Bits	Type	Reset	Description
31:00	RWC	0h	**Interrupt Vector Mask Clear (IVMC):** This field is bit significant. If a '1' is written to a bit, then the corresponding interrupt vector is unmasked. Writing a '0' to a bit has no effect. When read, this field returns the current interrupt mask value within the controller (not the value of this property). If a bit has a value of a '1', then the corresponding interrupt vector is masked. If a bit has a value of '0', then the corresponding interrupt vector is not masked.

3.1.3.5 Offset 14h: CC – Controller Configuration

This property modifies settings for the controller. Host software shall set the Arbitration Mechanism Selected (CC.AMS), the Memory Page Size (CC.MPS), and the I/O Command Set Selected (CC.CSS) to valid values prior to enabling the controller by setting CC.EN to '1'. Attempting to create an I/O queue before initializing the I/O Completion Queue Entry Size (CC.IOCQES) and the I/O Submission Queue Entry Size (CC.IOSQES) should cause a controller to abort a Create I/O Completion Queue command or a Create I/O Submission Queue command with a status code of Invalid Queue Size.

Figure 46: Offset 14h: CC – Controller Configuration

<table>
<tr><th>Bits</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:25</td><td>RO</td><td>0h</td><td>Reserved</td></tr>
<tr><td>24</td><td>RW/RO</td><td>0b</td><td>Controller Ready Independent of Media Enable (CRIME): This field controls the controller ready mode. The controller ready mode is determined by the state of this bit at the time the controller is enabled by transitioning the CC.EN bit from '0' to '1'.

If the CAP.CRMS field is set to 11b, then this bit is RW. If the CAP.CRMS field is not set to 11b, then this bit is RO and shall be cleared to '0'. Refer to sections 3.5.3 and 3.5.4 for more detail.

Changing the value of this field may cause a change in the time reported in the CAP.TO field. Refer to the definition of CAP.TO for more details.

<table><tr><th>Value</th><th>Definition</th></tr><tr><td>0b</td><td>Enabling the controller (i.e., CC.EN transitions from '0' to '1') when this bit is cleared to '0' enables Controller Ready With Media mode.</td></tr><tr><td>1b</td><td>Enabling the controller when this bit is set to '1' enables Controller Ready Independent of Media mode.</td></tr></table></td></tr>
<tr><td>23:20</td><td>RW/RO</td><td>0h</td><td>I/O Completion Queue Entry Size (IOCQES): This field defines the I/O completion queue entry size that is used for the selected I/O Command Set(s). The required and maximum values for this field are specified in the CQES field in the Identify Controller data structure in Figure 275 for each I/O Command Set. The value is in bytes and is specified as a power of two (2^n).

If any I/O Completion Queues exist, then write operations that change the value in this field produce undefined results.

If the controller does not support I/O queues, then this field shall be read-only with a value of 0h.

For Discovery controllers, this property is reserved.</td></tr>
</table>

Figure 46: Offset 14h: CC – Controller Configuration

<table>
<tr><th>Bits</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>19:16</td><td>RW/RO</td><td>0h</td><td>I/O Submission Queue Entry Size (IOSQES): This field defines the I/O submission queue entry size that is used for the selected I/O Command Set(s). The required and maximum values for this field are specified in the SQES field in the Identify Controller data structure in Figure 275 for each I/O Command Set. The value is in bytes and is specified as a power of two (2^n).

If any I/O Submission Queues exist, then write operations that change the value in this field produce undefined results.

If the controller does not support I/O queues, then this field shall be read-only with a value of 0h.

For Discovery controllers, this property is reserved.</td></tr>
<tr><td>15:14</td><td>RW</td><td>00b</td><td>Shutdown Notification (SHN): This field is used to initiate a controller shutdown when a power down condition is expected. For a normal controller shutdown, it is expected that the controller is given time to process the controller shutdown. For an abrupt shutdown, the host may not wait for the controller shutdown to complete before power is lost.

The controller shutdown notification values are defined as:

<table><tr><th>Value</th><th>Definition</th></tr><tr><td>00b</td><td>No notification; no effect</td></tr><tr><td>01b</td><td>Normal shutdown notification</td></tr><tr><td>10b</td><td>Abrupt shutdown notification</td></tr><tr><td>11b</td><td>Reserved</td></tr></table>
This field should be written by host software prior to any power down condition and prior to any change of the PCI power management state. It is recommended that this field also be written prior to a warm reset (refer to the PCI Express specification). To determine when the controller shutdown processing is complete, refer to CSTS.ST and CSTS.SHST. Refer to sections 3.6.1 and 3.6.2 for additional shutdown processing details.

Other fields in this property (including the EN bit) may be modified as part of updating this field to 01b or 10b to initiate a controller shutdown. If the EN bit is cleared to ‘0’ such that the EN bit transitions from ‘1’ to ‘0’, then both a Controller Reset and a controller shutdown occur.

If an NVM Subsystem Shutdown is in progress or is being reported as completed (i.e., CSTS.ST is set to ‘1’, and CSTS.SHST is set to 01b or 10b), then writes to this field modify the field value but have no effect. Refer to section 3.6.3 for details.</td></tr>
<tr><td>13:11</td><td>RW</td><td>000b</td><td>Arbitration Mechanism Selected (AMS): This field selects the arbitration mechanism to be used. This value shall only be changed when CC.EN is cleared to ‘0’. Host software shall only set this field to supported arbitration mechanisms indicated in CAP.AMS. If this field is set to an unsupported value, the behavior is undefined.

For Discovery controllers, this value shall be cleared to 0h.

<table><tr><th>Value</th><th>Definition</th></tr><tr><td>000b</td><td>Round Robin</td></tr><tr><td>001b</td><td>Weighted Round Robin with Urgent Priority Class</td></tr><tr><td>010b to 110b</td><td>Reserved</td></tr><tr><td>111b</td><td>Vendor Specific</td></tr></table></td></tr>
</table>

Figure 46: Offset 14h: CC – Controller Configuration

<table>
<tr><th>Bits</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>10:07</td><td>RW</td><td>0h</td><td>Memory Page Size (MPS): This field indicates the host memory page size. The memory page size is (2 ^ (12 + MPS)). Thus, the minimum host memory page size is 4 KiB and the maximum host memory page size is 128 MiB. The value set by host software shall be a supported value as indicated by the CAP.MPSMAX and CAP.MPSMIN fields. This field describes the value used for PRP entry size. This field shall only be modified when CC.EN is cleared to ‘0’.

For Discovery controllers this property shall be cleared to 0h.</td></tr>
<tr><td>06:04</td><td>RW</td><td>000b</td><td>I/O Command Set Selected (CSS): This field specifies the I/O Command Set or Sets that are selected. This field shall only be changed when the controller is disabled (i.e., CC.EN is cleared to ‘0’). The I/O Command Set or Sets that are selected shall be used for all I/O Submission Queues.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>
<table>
<tr><th>CAP.CSS bit 0</th><th>Description</th></tr>
<tr><td>1</td><td>NVM Command Set</td></tr>
<tr><td>0</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>001b to 101b</td><td>Reserved</td></tr>
<tr><td>110b</td><td>
<table>
<tr><th>CAP.CSS bit 6</th><th>Description</th></tr>
<tr><td>1</td><td>All Supported I/O Command Sets

The I/O Command Sets that are supported are reported in the Identify I/O Command Set data structure (refer to section 5.17.2.21).</td></tr>
<tr><td>0</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>111b</td><td>
<table>
<tr><th>CAP.CSS bit 7</th><th>Description</th></tr>
<tr><td>1</td><td>Admin Command Set only

I/O Command Set and I/O Command Set specific Admin commands are not supported. Any I/O Command Set specific Admin command submitted on the Admin Submission Queue is aborted with a status code of Invalid Command Opcode.</td></tr>
<tr><td>0</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>
For Discovery controllers, this property shall be cleared to 000b.</td></tr>
<tr><td>03:01</td><td>RO</td><td>000b</td><td>Reserved</td></tr>
</table>

Figure 46: Offset 14h: CC – Controller Configuration

Bits	Type	Reset	Description
00	RW	0b	**Enable (EN):** When set to '1', then the controller shall process commands. When cleared to '0', then the controller shall not process commands nor post completion queue entries to Completion Queues. When this bit transitions from '1' to '0', the controller is reset (i.e., a Controller Reset). That reset deletes all I/O Submission Queues and I/O Completion Queues, resets the Admin Submission Queue and Completion Queue, and brings the hardware to an idle state. That reset does not affect transport specific state (e.g. PCI Express registers including MMIO MSI-X registers), nor the Admin Queue properties (AQA, ASQ, or ACQ). All other controller properties defined in this section and internal controller state (e.g., Feature values defined in section 5.27.1 that are not persistent across power states) are reset to their default values. The controller shall ensure that there is no data loss for commands that have had corresponding completion queue entries posted to an I/O Completion Queue prior to that Controller Reset. Refer to section 3.6. When this bit is cleared to '0', the CSTS.RDY bit is cleared to '0' by the controller once the controller is ready to be re-enabled. When this bit is set to '1', the controller sets CSTS.RDY to '1' when it is ready to process commands. CSTS.RDY may be set to '1' before namespace(s) are ready to be accessed. Setting this bit from a '0' to a '1' when CSTS.RDY is a '1' or clearing this bit from a '1' to a '0' when CSTS.RDY is cleared to '0' has undefined results. The Admin Queue properties (AQA, ASQ, and ACQ) are only allowed to be modified when this bit is cleared to '0'. If an NVM Subsystem Shutdown is in progress or is completed (i.e., CSTS.ST is set to '1', and CSTS.SHST is set to 01b or 10b), then writes to this field modify the field value but have no effect. Refer to section 3.6.3 for details.

3.1.3.6 Offset 1Ch: CSTS – Controller Status

Figure 47: Offset 1Ch: CSTS – Controller Status

Bits	Type	Reset[1]	Description
31:07	RO	0h	Reserved
06	RO	Impl Spec	**Shutdown Type (ST):** When CSTS.SHST is set to a non-zero value, then this bit indicates the type of shutdown reported by CSTS.SHST. If this bit is set to '1', then CSTS.SHST is reporting the state of an NVM Subsystem Shutdown. If this bit is cleared to '0', then CSTS.SHST is reporting the state of a controller shutdown. If CSTS.SHST is cleared to 00b, then this bit is ignored.
05	RO	0b	**Processing Paused (PP):** This bit indicates whether the controller is processing commands. If this bit is cleared to '0', then the controller is processing commands normally. If this bit is set to '1', then the controller has temporarily stopped processing commands in order to handle an event (e.g., firmware activation). This bit is only valid when CC.EN is set to '1'.
04	RWC	HwInit	**NVM Subsystem Reset Occurred (NSSRO):** The initial value of this bit is set to '1' if the last occurrence of an NVM Subsystem Reset (refer to section 3.7.1) occurred while power was applied to the domain. The initial value of this bit is cleared to '0' following an NVM Subsystem Reset due to application of power to the domain. This bit is only valid if the controller supports the NVM Subsystem Reset feature defined in section 3.7.1 as indicated by CAP.NSSRS set to '1'. The reset value of this bit is cleared to '0' if an NVM Subsystem Reset causes activation of a new firmware image in the domain.

Figure 47: Offset 1Ch: CSTS – Controller Status

<table>
<tr><th>Bits</th><th>Type</th><th>Reset[1]</th><th>Description</th></tr>
<tr><td>03:02</td><td>RO</td><td>00b</td><td>Shutdown Status (SHST): This field indicates the status of shutdown processing that is initiated by the host setting the CC.SHN field, the host setting the NSSC property, or a Management Endpoint has processed an NVMe-MI Shutdown command (refer to the NVM Express Management Interface specification).

The shutdown status values are defined as:

<table><tr><th>Value</th><th>Definition</th></tr><tr><td>00b</td><td>Normal operation (no shutdown has been requested)</td></tr><tr><td>01b</td><td>Shutdown processing occurring</td></tr><tr><td>10b</td><td>Shutdown processing complete</td></tr><tr><td>11b</td><td>Reserved</td></tr></table>
If this field is set to 01b, then:

• an NVM Subsystem Reset aborts a controller shutdown and an NVM Subsystem Shutdown; and
• any other type of Controller Level Reset:
o may or may not abort a controller shutdown; and
o shall not abort an NVM Subsystem Shutdown.

If this field is set to 01b when a Controller Level Reset is initiated and the shutdown is not aborted, then this field transitions to 00b on the reset and then to 01b to indicate the shutdown is still in progress and host software may or may not observe this transition.

If CSTS.ST is cleared to ‘0’ and this field is set to 10b, then
• If CC.EN is set to ‘1’, to start executing commands on the controller after a shutdown operation (CSTS.SHST set to 10b), a Controller Reset (CC.EN cleared to ‘0’) is required. If host software submits commands to the controller without issuing a Controller Reset, the behavior is undefined; and
• If CC.EN is cleared to ‘0’, to start executing commands on the controller:
o a Controller Level Reset is required; or
o CC.EN is required to be set to ‘1’ and CC.SHN is required to be cleared to 00b with the same write to the CC property.

If CSTS.ST is set to ‘1’ and this field is set to 10b, then an NVM Subsystem Reset is required to start executing commands.

Refer to section 3.6.3 on the reset behavior of this field when CAP.CPS is set to 10b or 11b.</td></tr>
<tr><td>01</td><td>RO</td><td>HwInit</td><td>Controller Fatal Status (CFS): This bit is set to ’1’ when a fatal controller error occurred that could not be communicated in the appropriate Completion Queue. This bit is cleared to ‘0’ when a fatal controller error has not occurred. Refer to section 9.5.

The reset value of this bit is set to '1' when a fatal controller error is detected during controller initialization.</td></tr>
<tr><td>00</td><td>RO</td><td>0b</td><td>Ready (RDY): This bit is set to ‘1’ when the controller is ready to process submission queue entries after CC.EN is set to ‘1’. This bit shall be cleared to ‘0’ when CC.EN is cleared to ‘0’ once the controller is ready to be re-enabled. Commands should not be submitted to the controller until this bit is set to ‘1’ after the CC.EN bit is set to ‘1’. Failure to follow this recommendation produces undefined results. Refer to the definition of CAP.TO, sections 3.5.3, and 3.5.4 for timing information related to this field.

If an NVM Subsystem Shutdown has completed that affects this controller (i.e., CSTS.ST is set to ‘1’ and CSTS.SHST is set to 10b), then an NVM Subsystem Reset is required before this bit is allowed to be set to ‘1’. Refer to section 3.6.3.</td></tr>
<tr><td colspan="4">NOTE:
1. During a Controller Level Reset, the field values may transition to values other than the reset value prior to indicating the reset value.</td></tr>
</table>

3.1.3.7 Offset 20h: NSSR – NVM Subsystem Reset

This optional property provides host software with the capability to initiate an NVM Subsystem Reset. Support for this property is indicated by the state of the NVM Subsystem Reset Supported (CAP.NSSRS) field. If the property is not supported, then the address range occupied by the property is reserved. Refer to section 3.7.1.

Figure 48: Offset 20h: NSSR – NVM Subsystem Reset

Bits	Type	Reset	Description
31:00	RW	0h	**NVM Subsystem Reset Control (NSSRC):** A write of the value 4E564D65h ("NVMe") to this field initiates an NVM Subsystem Reset. A write of any other value has no functional effect on the operation of the NVM subsystem. This field shall return the value 0h when read.

3.1.3.8 Offset 24h: AQA – Admin Queue Attributes

This property defines the attributes for the Admin Submission Queue and Admin Completion Queue. The Queue Identifier for the Admin Submission Queue and Admin Completion Queue is 0h. The Admin Submission Queue's priority is determined by the arbitration mechanism selected, refer to section 3.4.4. The Admin Submission Queue and Admin Completion Queue are required to be in physically contiguous memory.

Note: It is recommended that UEFI be used during boot operations. In low memory environments (like Option ROMs in legacy BIOS environments) there may not be sufficient available memory to allocate the necessary Submission and Completion Queues. In these types of conditions, low memory operation of the controller is vendor specific.

Figure 49: Offset 24h: AQA – Admin Queue Attributes

Bits	Type	Reset	Description
31:28	RO	0h	Reserved
27:16	RW	0h	**Admin Completion Queue Size (ACQS):** Defines the size of the Admin Completion Queue in entries. Refer to section 3.3.3.2.2. Enabling a controller while this field is cleared to 0h produces undefined results. The minimum size of the Admin Completion Queue is two entries. The maximum size of the Admin Completion Queue is 4,096 entries. This is a 0's based value.
15:12	RO	0h	Reserved
11:00	RW	0h	**Admin Submission Queue Size (ASQS):** Defines the size of the Admin Submission Queue in entries. Refer to section 3.3.3.2.2. Enabling a controller while this field is cleared to 0h produces undefined results. The minimum size of the Admin Submission Queue is two entries. The maximum size of the Admin Submission Queue is 4,096 entries. This is a 0's based value.

3.1.3.9 Offset 28h: ASQ – Admin Submission Queue Base Address

This property defines the base memory address of the Admin Submission Queue.

Figure 50: Offset 28h: ASQ – Admin Submission Queue Base Address

Bits	Type	Reset	Description
63:12	RW	Impl Spec	**Admin Submission Queue Base (ASQB):** This field specifies the 52 most significant bits of the 64-bit physical address for the Admin Submission Queue. This address shall be memory page aligned (based on the value in CC.MPS). All Admin commands, including creation of I/O Submission Queues and I/O Completions Queues shall be submitted to this queue. For the definition of Submission Queues, refer to section 4.1.
11:00	RO	0h	Reserved

3.1.3.10 Offset 30h: ACQ – Admin Completion Queue Base Address

This property defines the base memory address of the Admin Completion Queue.

Figure 51: Offset 30h: ACQ – Admin Completion Queue Base Address

Bit	Type	Reset	Description
63:12	RW	Impl Spec	**Admin Completion Queue Base (ACQB):** This field specifies the 52 most significant bits of the 64-bit physical address for the Admin Completion Queue. This address shall be memory page aligned (based on the value in CC.MPS). All completion queue entries for the commands submitted to the Admin Submission Queue shall be posted to this Completion Queue. This queue is always associated with interrupt vector 0. For the definition of Completion Queues, refer to section 4.1.
11:00	RO	0h	Reserved

3.1.3.11 Offset 38h: CMBLOC – Controller Memory Buffer Location

This optional property defines the location of the Controller Memory Buffer (refer to section 8.1). If the controller does not support the Controller Memory Buffer (CAP.CMBS), this property is reserved. If the controller supports the Controller Memory Buffer and CMBMSC.CRE is cleared to '0', this property shall be cleared to 0h.

Figure 52: Offset 38h: CMBLOC – Controller Memory Buffer Location

Bits	Type	Reset	Description
31:12	RO	Impl Spec	**Offset (OFST):** Indicates the offset of the Controller Memory Buffer in multiples of the Size Unit specified in CMBSZ.
11:09	RO	000b	Reserved
08	RO	Impl Spec	**CMB Queue Dword Alignment (CQDA):** If this bit is set to '1', CDW11.PC is set to '1'; and the address pointer specifies Controller Memory Buffer, then the address pointer in a Create I/O Submission Queue command (refer to Figure 159) or a Create I/O Completion Queue command (refer to Figure 155) shall be Dword aligned. If this bit is cleared to '0', then the I/O Submission Queues and I/O Completion Queues contained in the Controller Memory Buffer are aligned as defined by the PRP1 field of a Create I/O Submission Queue command (refer to Figure 159) or a Create I/O Completion Queue command (refer to Figure 155).
07	RO	Impl Spec	**CMB Data Metadata Mixed Memory Support (CDMMMS):** If this bit is set to '1', then the restriction on data and metadata use of Controller Buffer Memory by a command as defined in section 8.5 is not enforced. If this bit is cleared to '0', then the restriction on data and metadata use of Controller Buffer Memory by a command as defined in section 8.5 is enforced.
06	RO	Impl Spec	**CMB Data Pointer and Command Independent Locations Support (CDPCILS):** If this bit is set to '1', then the restriction that the PRP Lists and SGLs shall not be located in the Controller Buffer Memory if the command that they are associated with is not located in the Controller Buffer Memory is not enforced (refer to section 8.5). If this bit is cleared to '0', then that restriction is enforced.
05	RO	Impl Spec	**CMB Data Pointer Mixed Locations Support (CDPMLS):** If this bit is set to '1', then the restriction that for a particular PRP List or SGL associated with a single command, all memory that is associated with that particular PRP List or SGL shall reside in either the Controller Memory Buffer or outside the Controller Memory Buffer, is not enforced (refer to section 8.5). If this bit is cleared to '0', then that restriction is enforced.
04	RO	Impl Spec	**CMB Queue Physically Discontiguous Support (CQPDS):** If this bit is set to '1', then the restriction that for all queues in the Controller Memory Buffer, the queue shall be physically contiguous, is not enforced (refer to section 8.5). If this bit is cleared to '0', then that restriction is enforced.
03	RO	Impl Spec	**CMB Queue Mixed Memory Support (CQMMS):** If this bit is set to '1', then for a particular queue placed in the Controller Memory Buffer, the restriction that all memory associated with that queue shall reside in the Controller Memory Buffer is not enforced (refer to section 8.5). If this bit is cleared to '0', then that requirement is enforced.

Figure 52: Offset 38h: CMBLOC – Controller Memory Buffer Location

Bits	Type	Reset	Description
02:00	RO	Impl Spec	**Base Indicator Register (BIR):** Indicates the Base Address Register (BAR) that contains the Controller Memory Buffer. For a 64-bit BAR, the BAR for the least significant 32-bits of the address is specified. Values 000b, 010b, 011b, 100b, and 101b are valid. The address specified by the BAR shall be 4 KiB aligned.

3.1.3.12 Offset 3Ch: CMBSZ – Controller Memory Buffer Size

This optional property defines the size of the Controller Memory Buffer (refer to section 8.1). If the controller does not support the Controller Memory Buffer feature or if the controller supports the Controller Memory Buffer (CAP.CMBS) and CMBMSC.CRE is cleared to '0', then this property shall be cleared to 0h.

Figure 53: Offset 3Ch: CMBSZ – Controller Memory Buffer Size

<table>
<tr><th>Bits</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:12</td><td>RO</td><td>Impl Spec</td><td>Size (SZ): Indicates the size of the Controller Memory Buffer available for use by the host. The size is in multiples of the Size Unit. If the Offset + Size exceeds the length of the indicated BAR, the size available to the host is limited by the length of the BAR.</td></tr>
<tr><td>11:08</td><td>RO</td><td>Impl Spec</td><td>Size Units (SZU): Indicates the granularity of the Size field.
<table>
<tr><th>Value</th><th>Granularity</th></tr>
<tr><td>0h</td><td>4 KiB</td></tr>
<tr><td>1h</td><td>64 KiB</td></tr>
<tr><td>2h</td><td>1 MiB</td></tr>
<tr><td>3h</td><td>16 MiB</td></tr>
<tr><td>4h</td><td>256 MiB</td></tr>
<tr><td>5h</td><td>4 GiB</td></tr>
<tr><td>6h</td><td>64 GiB</td></tr>
<tr><td>7h to Fh</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>07:05</td><td>RO</td><td>000b</td><td>Reserved</td></tr>
<tr><td>04</td><td>RO</td><td>Impl Spec</td><td>Write Data Support (WDS): If this bit is set to '1', then the controller supports data and metadata in the Controller Memory Buffer for commands that transfer data from the host to the controller (e.g., Write). If this bit is cleared to '0', then data and metadata for commands that transfer data from the host to the controller shall not be transferred to the Controller Memory Buffer.</td></tr>
<tr><td>03</td><td>RO</td><td>Impl Spec</td><td>Read Data Support (RDS): If this bit is set to '1', then the controller supports data and metadata in the Controller Memory Buffer for commands that transfer data from the controller to the host (e.g., Read). If this bit is cleared to '0', then data and metadata for commands that transfer data from the controller to the host shall not be transferred from the Controller Memory Buffer.</td></tr>
<tr><td>02</td><td>RO</td><td>Impl Spec</td><td>PRP SGL List Support (LISTS): If this bit is set to '1', then the controller supports PRP Lists in the Controller Memory Buffer. If this bit is set to '1' and SGLs are supported by the controller, then the controller supports Scatter Gather Lists in the Controller Memory Buffer. If this bit is set to '1', then the Submission Queue Support bit shall be set to '1'. If this bit is cleared to '0', then PRP Lists and SGLs shall not be placed in the Controller Memory Buffer.</td></tr>
<tr><td>01</td><td>RO</td><td>Impl Spec</td><td>Completion Queue Support (CQS): If this bit is set to '1', then the controller supports Admin and I/O Completion Queues in the Controller Memory Buffer. If this bit is cleared to '0', then Completion Queues shall not be placed in the Controller Memory Buffer.</td></tr>
<tr><td>00</td><td>RO</td><td>Impl Spec</td><td>Submission Queue Support (SQS): If this bit is set to '1', then the controller supports Admin and I/O Submission Queues in the Controller Memory Buffer. If this bit is cleared to '0', then Submission Queues shall not be placed in the Controller Memory Buffer.</td></tr>
</table>

3.1.3.13 Offset 40h: BPINFO – Boot Partition Information

This optional property defines the characteristics of Boot Partitions (refer to section 8.2). If the controller does not support the Boot Partitions feature, then this property shall be cleared to 0h.

Figure 54: Offset 40h: BPINFO – Boot Partition Information

<table>
<tr><th>Bits</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31</td><td>RO</td><td>Impl Spec</td><td>Active Boot Partition ID (ABPID): This bit indicates the identifier of the active Boot Partition.</td></tr>
<tr><td>30:26</td><td>RO</td><td>0h</td><td>Reserved</td></tr>
<tr><td>25:24</td><td>RO</td><td>00b</td><td>Boot Read Status (BRS): This field indicates the status of Boot Partition read operations initiated by the host writing to the BPRSEL.BPID field. Refer to section 8.2.

The boot read status values are defined as:

<table><tr><th>Value</th><th>Definition</th></tr><tr><td>00b</td><td>No Boot Partition read operation requested</td></tr><tr><td>01b</td><td>Boot Partition read in progress</td></tr><tr><td>10b</td><td>Boot Partition read completed successfully</td></tr><tr><td>11b</td><td>Error completing Boot Partition read</td></tr></table>
If host software writes the BPRSEL.BPID field, this field transitions to 01b. After successfully completing a Boot Partition read operation (i.e., transferring the contents to the boot memory buffer), the controller sets this field to 10b. If there is an error completing a Boot Partition read operation, this field is set to 11b, and the contents of the boot memory buffer are undefined.</td></tr>
<tr><td>23:15</td><td>RO</td><td>0h</td><td>Reserved</td></tr>
<tr><td>14:00</td><td>RO</td><td>Impl Spec</td><td>Boot Partition Size (BPSZ): This field defines the size of each Boot Partition in multiples of 128 KiB. Both Boot Partitions are the same size.</td></tr>
</table>

3.1.3.14 Offset 44h: BPRSEL – Boot Partition Read Select

This optional property is used to initiate the transfer of a data in the Boot Partition (refer to section 8.2) from the controller to the host. If the controller does not support the Boot Partitions feature, then this property shall be cleared to 0h.

If the host attempts to read beyond the end of a Boot Partition (i.e., the Boot Partition Read Offset plus Boot Partition Read Size, is greater than the Boot Partition Size in bytes), the controller shall not transfer data and report an error in the BPINFO.BRS field.

Figure 55: Offset 44h: BPRSEL – Boot Partition Read Select

Bits	Type	Reset	Description
31	RW	0b	**Boot Partition Identifier (BPID):** This bit specifies the Boot Partition identifier for the Boot Partition read operation.
30	RO	0b	Reserved
29:10	RW	0h	**Boot Partition Read Offset (BPROF):** This field selects the offset into the Boot Partition, in 4 KiB units, that the controller copies into the Boot Partition Memory Buffer.
09:00	RW	0h	**Boot Partition Read Size (BPRSZ):** This field selects the read size in multiples of 4 KiB to copy into the Boot Partition Memory Buffer.

3.1.3.15 Offset 48h: BPMBL – Boot Partition Memory Buffer Location

This optional property specifies the memory buffer that is used as the destination for data when a Boot Partition is read (refer to section 8.2). If the controller does not support the Boot Partitions feature, then this property shall be cleared to 0h.

Figure 56: Offset 48h: BPMBL – Boot Partition Memory Buffer Location

Bits	Type	Reset	Description
63:12	RW	0h	**Boot Partition Memory Buffer Base Address (BMBBA):** This field specifies the 52 most significant bits of the 64-bit physical address for the Boot Partition Memory Buffer.

Figure 56: Offset 48h: BPMBL – Boot Partition Memory Buffer Location

Bits	Type	Reset	Description
11:00	RO	0h	Reserved

3.1.3.16 Offset 50h: CMBMSC – Controller Memory Buffer Memory Space Control

This optional property specifies how the controller references the Controller Memory Buffer with host-supplied addresses. If the controller supports the Controller Memory Buffer (CAP.CMBS), this property is mandatory. Otherwise, this property is reserved.

This property shall be reset by neither Controller Reset nor Function Level Reset, but it shall be reset by all other Controller Level Resets.

Figure 57: Offset 50h: CMBMSC – Controller Memory Buffer Memory Space Control

Bits	Type	Reset	Description
63:12	RW	0h	**Controller Base Address (CBA):** This field specifies the 52 most significant bits of the 64-bit base address for the Controller Memory Buffer's controller address range. The Controller Memory Buffer's controller base address and its size determine its controller address range. The specified address shall be valid only under the following conditions: a) no part of the Controller Memory Buffer's controller address range is greater than $2^{64} - 1$; and b) if the Persistent Memory Region's controller memory space is enabled, then the Controller Memory Buffer's controller address range does not overlap the Persistent Memory Region's controller address range.
11:02	RO	0h	Reserved
01	RW	0b	**Controller Memory Space Enable (CMSE):** This bit specifies whether addresses supplied by the host are permitted to reference the Controller Memory Buffer. If CMBSMSC.CRE is cleared to '0' this bit has no effect, and the Controller Memory Buffer's controller memory space is not enabled. If this bit is set to '1' and the controller base address is valid, then the Controller Memory Buffer's controller memory space is enabled. Otherwise, the controller memory space is not enabled. If the Controller Memory Buffer's controller memory space is enabled, then addresses supplied by the host that fall within the Controller Memory Buffer's controller address range shall reference the Controller Memory Buffer. If the Controller Memory Buffer's controller memory space is not enabled, then no address supplied by the host shall reference the Controller Memory Buffer. Instead, such addresses shall reference memory spaces other than the Controller Memory Buffer.
00	RW	0b	**Capabilities Registers Enabled (CRE):** This bit specifies whether the CMBLOC and CMBSZ properties are enabled. If this bit is set to '1', then CMBLOC is defined as shown in Figure 52 and CMBSZ is defined as shown in Figure 53. If this bit is cleared to '0', then CMBSZ and CMBLOC are cleared to 0h.

3.1.3.17 Offset 58h: CMBSTS – Controller Memory Buffer Status

This optional property indicates the status of the Controller Memory Buffer. If the controller supports the Controller Memory Buffer (CAP.CMBS), this property is mandatory. Otherwise, this property is reserved.

Figure 58: Offset 58h: CMBSTS – Controller Memory Buffer Status

Bits	Type	Reset	Description
31:01	RO	0h	Reserved

Figure 58: Offset 58h: CMBSTS – Controller Memory Buffer Status

Bits	Type	Reset	Description
00	RO	0b	**Controller Base Address Invalid (CBAI):** This bit indicates whether the controller has failed to enable the Controller Memory Buffer's controller memory space because CMBMSC.CBA is invalid. If CMBMSC.CRE and CMBMSC.CMSE are set to '1', and CMBMSC.CBA is invalid, this bit shall be set to '1'. Otherwise, this bit shall be cleared to '0'.

3.1.3.18 Offset 5Ch: CMBEBS – Controller Memory Buffer Elasticity Buffer Size

This optional property identifies to the host the size of the CMB elasticity buffer. A value of 0h in this property indicates to the host that no information regarding the presence or size of a CMB elasticity buffer is available.

Figure 59: Offset 5Ch: CMBEBS – Controller Memory Buffer Elasticity Buffer Size

Bits	Type	Reset	Description
31:8	RO	Impl Spec	**CMB Elasticity Buffer Size Base (CMBWBZ):** Indicates the size of the CMB elasticity buffer. The size of the CMB elasticity buffer is equal to the value in this field multiplied by the value specified by the CMB Elasticity Buffer Size Units field.
7:5	RO	0h	Reserved
4	RO	Impl Spec	**Read Bypass Behavior:** If a memory read does not conflict with any memory write in the CMB Elasticity Buffer (i.e., if the set of memory addresses specified by a read is disjoint from the set of memory addresses specified by all writes in the CMB Elasticity Buffer), and this bit is: a) set to '1', then memory reads not conflicting with memory writes in the CMB Elasticity Buffer shall bypass those memory writes; and b) cleared to '0', then memory reads not conflicting with memory writes in the CMB Elasticity Buffer may bypass those memory writes.
3:0	RO	Impl Spec	**CMB Elasticity Buffer Size Units (CMBSZU):** Indicates the granularity of the CMB Elasticity Buffer Size field. Value: Granularity 0h: Bytes 1h: 1 KiB 2h: 1 MiB 3h: 1 GiB 4h – Fh: Reserved

3.1.3.19 Offset 60h: CMBSWTP – Controller Memory Buffer Sustained Write Throughput

This optional property identifies to the host the maximum CMB sustained write throughput. A value of 0h in this property indicates to the host that no information regarding the CMB sustained write throughput is available.

Figure 60: Offset 60h: CMBSWTP – Controller Memory Buffer Sustained Write Throughput

Bits	Type	Reset	Description
31:8	RO	Impl Spec	**CMB Sustained Write Throughput (CMBSWTV):** Indicates the sustained write throughput of the CMB at the maximum payload size specified by the applicable NVMe Transport binding specification (e.g. the PCIe TLP payload size, as specified in the Max_Payload_Size (MPS) field of the PCIe Express Device Control (PXDC) register). The sustained write throughput of the CMB is equal to the value in this field multiplied by the units specified by the CMB Sustained Write Throughput Units field.
7:4	RO	0h	Reserved

<table>
<tr>
<td>3:0</td>
<td>RO</td>
<td>Impl Spec</td>
<td>CMB Sustained Write Throughput Units (CMBSWTU): Indicates the granularity of the CMB Sustained Write Throughput field.
<table>
<tr><th>Value</th><th>Granularity</th></tr>
<tr><td>0h</td><td>Bytes/second</td></tr>
<tr><td>1h</td><td>1 KiB/second</td></tr>
<tr><td>2h</td><td>1 MiB/second</td></tr>
<tr><td>3h</td><td>1 GiB/second</td></tr>
<tr><td>4h – Fh</td><td>Reserved</td></tr>
</table>
</td>
</tr>
</table>

3.1.3.20 Offset 64h: NSSD – NVM Subsystem Shutdown

This optional property provides host software with the capability to initiate a normal or abrupt NVM subsystem shutdown.

Support for this property is indicated by the state of the NVM Subsystem Shutdown Supported (CAP.NSSS) field. If the property is not supported, then the address range occupied by the register is reserved.

Figure 61: Offset 64h: NSSD – NVM Subsystem Shutdown

Bits	Type	Reset	Description
31:00	RW	0h	**NVM Subsystem Shutdown Control (NSSC):** A write of the value 4E726D6Ch ("Nrml") to this field initiates a normal NVM Subsystem Shutdown on every controller: • in the domain associated with the controller when CAP.CPS is set to 10b as specified in section 3.6.3.1; or • in the NVM subsystem when CAP.CPS is set to 11b in the NVM subsystem as specified in section 3.6.3.2. A write of the value 41627074h ("Abpt") to this field initiates an abrupt NVM subsystem shutdown on every controller: • in the domain associated with the controller when CAP.CPS is set to 10b as specified in section 3.6.3.1; or • in the NVM subsystem when CAP.CPS is set to 11b in the NVM subsystem as specified in section 3.6.3.2. A write of any other value to this field has no functional effect on the operation of the NVM subsystem. This field shall return the value 0h when read.

3.1.3.21 Offset 68h: CRTO – Controller Ready Timeouts

This property indicates the controller ready timeout values. This property is mandatory for controllers compliant with NVM Express Base Specification revision 2.0 and later.

Figure 62: Offset 68h: CRTO – Controller Ready Timeouts

Bits	Type	Reset	Description
31:16	RO	Impl Spec	**Controller Ready Independent of Media Timeout (CRIMT):** If the CAP.CRMS.CRIMS bit is cleared to '0', then this field is not applicable and shall be cleared to 0h. If the CAP.CRMS.CRIMS bit is set to '1', then this field contains the worst-case time that host software should wait after CC.EN transitions from '0' to '1' for the controller to become ready and be able to successfully process all commands that do not access attached namespaces and Admin commands that do not require access to media when the controller is in Controller Ready Independent of Media mode (i.e., the CC.CRIME bit is set to '1'). Attached namespaces and media required to process Admin commands may or may not be ready within this time period (refer to section 3.5.3, section 3.5.4, and Figure 104). This worst-case time may be experienced after events such as an abrupt shutdown or activation of a new firmware image; typical times are expected to be much shorter. This field is in 500 millisecond units. The value of this field should not exceed FFh (i.e., 127.5 seconds).
15:0	RO	Impl Spec	**Controller Ready With Media Timeout (CRWMT):** This field contains the worst-case time that host software should wait after CC.EN transitions from '0' to '1' for: a) the controller to become ready and be able to successfully process all commands; and b) all attached namespaces and media required to process Admin commands to become ready, independent of which ready mode (refer to CC.CRIME) the controller is in (refer to section 3.5.3 and section 3.5.4). This worst-case time may be experienced after events such as an abrupt shutdown or activation of a new firmware image; typical times are expected to be much shorter. This field is in 500 millisecond units. The value of this field shall be greater than or equal to the value of the CRTO.CRIMT field and may be significantly larger than the value of the CRTO.CRIMT field.

3.1.3.22 Offset E00h: PMRCAP – Persistent Memory Region Capabilities

This property indicates capabilities of the Persistent Memory Region. If the controller does not support the Persistent Memory Region feature, then this property shall be cleared to 0h.

Figure 63: Offset E00h: PMRCAP – Persistent Memory Region Capabilities

Bits	Type	Reset	Description
31:25	RO	0h	Reserved
24	RO	Impl Spec	**Controller Memory Space Supported (CMSS):** If set to '1', this bit indicates that addresses supplied by the host are permitted to reference the Persistent Memory Region only if the host has enabled the Persistent Memory Region's controller memory space. If the controller supports referencing the Persistent Memory Region with host-supplied addresses, this bit shall bet set to '1'. Otherwise, this bit shall be cleared to '0'.
23:16	RO	Impl Spec	**Persistent Memory Region Timeout (PMRTO):** This field contains the minimum amount of time that host software should wait for the Persistent Memory Region to become ready or not ready after PMRCTL.EN is modified. The time in this field is expressed in Persistent Memory Region time units (refer to PMRCAP.PMRTU).
15:14	RO	00b	Reserved

Figure 63: Offset E00h: PMRCAP – Persistent Memory Region Capabilities

<table>
<tr><th>Bits</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>13:10</td><td>RO</td><td>Impl Spec</td><td>Persistent Memory Region Write Barrier Mechanisms (PMRWBM): This field lists mechanisms that may be used to ensure that previous writes to the Persistent Memory Region have completed and are persistent when the Persistent Memory Region is ready and operating normally. A bit in this field is set to ‘1’ if the corresponding mechanism to ensure persistence is supported. A bit in this field is cleared to ‘0’ if the corresponding mechanism to ensure persistence is not supported.

At least one bit in this field shall be set to ‘1’.

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>0</td><td>The completion of a memory read from any Persistent Memory Region address ensures that all prior writes to the Persistent Memory Region have completed and are persistent.</td></tr>
<tr><td>1</td><td>The completion of a read to the PMRSTS property shall ensure that all prior writes to the Persistent Memory Region have completed and are persistent.</td></tr>
<tr><td>3:2</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>9:8</td><td>RO</td><td>Impl Spec</td><td>Persistent Memory Region Time Units (PMRTU): Indicates Persistent Memory Region time units.

<table>
<tr><th>Value</th><th>Persistent Memory Region Time Units</th></tr>
<tr><td>00b</td><td>500 milliseconds</td></tr>
<tr><td>01b</td><td>minutes</td></tr>
<tr><td>10b to 11b</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>7:5</td><td>RO</td><td>Impl Spec</td><td>Base Indicator Register (BIR): This field indicates the Base Address Register (BAR) that specifies the address and size of the Persistent Memory Region. Values 010b, 011b, 100b, and 101b are valid.</td></tr>
<tr><td>4</td><td>RO</td><td>Impl Spec</td><td>Write Data Support (WDS): If this bit is set to ‘1’, then the controller supports data and metadata in the Persistent Memory Region for commands that transfer data from the host to the controller (e.g., Write). If this bit is cleared to ‘0’, then data and metadata for commands that transfer data from the host to the controller shall not be transferred to the Persistent Memory Region.

If PMRCAP.CMSS is cleared to ‘0’, this bit shall be cleared to ‘0’.</td></tr>
<tr><td>3</td><td>RO</td><td>Impl Spec</td><td>Read Data Support (RDS): If this bit is set to ‘1’, then the controller supports data and metadata in the Persistent Memory Region for commands that transfer data from the controller to the host (e.g., Read). If this bit is cleared to ‘0’, then all data and metadata for commands that transfer data from the controller to the host shall not be transferred from the Persistent Memory Region.

If PMRCAP.CMSS is cleared to ‘0’, this bit shall be cleared to ‘0’.</td></tr>
<tr><td>2:0</td><td>RO</td><td>000b</td><td>Reserved</td></tr>
</table>

3.1.3.23 Offset E04h: PMRCTL – Persistent Memory Region Control

This optional property controls the operation of the Persistent Memory Region. If the controller does not support the Persistent Memory Region feature, then this property shall be cleared to 0h.

Figure 64: Offset E04h: PMRCTL – Persistent Memory Region Control

Bits	Type	Reset	Description
31:1	RO	0h	Reserved
0	RW	0b	**Enable (EN):** When set to ‘1’, then the Persistent Memory Region is ready to process PCI Express memory read and write requests once PMRSTS.NRDY is cleared to ‘0’. When cleared to ‘0’, then the Persistent Memory Region is disabled and PMRSTS.NRDY shall be set to ‘1’ once the Persistent Memory Region is ready to be re-enabled.

3.1.3.24 Offset E08h: PMRSTS – Persistent Memory Region Status

This optional property provides the status of the Persistent Memory Region. If the controller does not support the Persistent Memory Region feature, then this property shall be cleared to 0h.

Figure 65: Offset E08h: PMRSTS – Persistent Memory Region Status

<table>
<tr><th>Bits</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:13</td><td>RO</td><td>0h</td><td>Reserved</td></tr>
<tr><td>12</td><td>RO</td><td>0b</td><td>Controller Base Address Invalid (CBAI): This field indicates whether the controller has failed to enable the Persistent Memory Region’s controller memory space because the controller 64-bit base address specified by PMRMSCU.CBA and PMRMSCL.CBA are invalid. If PMRCAP.CMSS is set to ‘1’, PMRMSCL.CMSE is set to ‘1’, and the controller 64-bit base address specified by PMRMSCU.CBA and PMRMSCL.CBA is invalid, this bit shall be set to ‘1’. Otherwise, this bit shall be cleared to ‘0’.</td></tr>
<tr><td>11:9</td><td>RO</td><td>000b</td><td>Health Status (HSTS): If the Persistent Memory Region is ready, then this field indicates the health status of the Persistent Memory Region. This field is always cleared to 000b when the Persistent Memory Region is not ready.

The health status values are defined as:

<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>Normal Operation: The Persistent Memory Region is operating normally.</td></tr>
<tr><td>001b</td><td>Restore Error: The Persistent Memory Region is operating normally and is persistent; however, the contents of the Persistent Memory Region may not have been restored correctly (i.e., may not contain the contents prior to the last power cycle, NVM Subsystem Reset, Controller Level Reset, or Persistent Memory Region disable).</td></tr>
<tr><td>010b</td><td>Read Only: The Persistent Memory Region is read only. PCI Express memory write requests do not update the Persistent Memory Region. PCI Express memory read requests to the Persistent Memory Region return correct data.</td></tr>
<tr><td>011b</td><td>Unreliable: The Persistent Memory Region has become unreliable. PCI Express memory reads may return invalid data or generate poisoned PCI Express TLP(s). Persistent Memory Region memory writes may not update memory or may update memory with undefined data. The Persistent Memory Region may also have become non-persistent.</td></tr>
<tr><td>100b to 111b</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>8</td><td>RO</td><td>0b</td><td>Not Ready (NRDY): This bit indicates if the Persistent Memory Region is ready for use. If this bit is cleared to ‘0’ and the PMRCTL.EN is set to ‘1’, then the Persistent Memory Region is ready to accept and process PCI Express memory read and write requests. If this bit is set to ‘1’ or the PMRCTL.EN bit is cleared to ‘0’, then the Persistent Memory Region is not ready to process PCI Express memory read and write requests.</td></tr>
<tr><td>7:0</td><td>RO</td><td>0h</td><td>Error (ERR): When the Persistent Memory Region is ready and operating normally, this field indicates whether previous memory writes to the Persistent Memory Region have completed without error. If this field is cleared to 0h, then previous writes to the Persistent Memory Region have completed without error and that the values written are persistent. A non-zero value in this field indicates the occurrence of an error that may have caused one or more of the previous writes to not have completed successfully. The meaning of any particular non-zero value is vendor specific.

Once this field takes on a non-zero value, it maintains a non-zero value until the PCI Function is reset.</td></tr>
</table>

3.1.3.25 Offset E0Ch: PMREBS – Persistent Memory Region Elasticity Buffer Size

This optional property identifies to the host the size of the PMR elasticity buffer. A value of 0h in this property indicates to the host that no information regarding the presence or size of a PMR elasticity buffer is available.

Figure 66: Offset E0Ch: PMREBS – Persistent Memory Region Elasticity Buffer Size

Bits	Type	Reset	Description
31:8	RO	Impl Spec	**PMR Elasticity Buffer Size Base (PMRWBZ):** Indicates the size of the PMR elasticity buffer. The actual size of the PMR elasticity buffer is equal to the value in this field multiplied by the value specified by the PMR Elasticity Buffer Size Units field.
7:5	RO	000b	Reserved
4	RO	Impl Spec	**Read Bypass Behavior:** If a memory read does not conflict with any memory write in the PMR Elasticity Buffer (i.e., if the set of memory addresses specified by a read is disjoint from the set of memory addresses specified by all writes in the PMR Elasticity Buffer), and this bit is: a) set to '1', then memory reads not conflicting with memory writes in the PMR Elasticity Buffer shall bypass those memory writes; and b) cleared to '0', then memory reads not conflicting with memory writes in the PMR Elasticity Buffer may bypass those memory writes.
3:0	RO	Impl Spec	**PMR Elasticity Buffer Size Units (PMRSZU):** Indicates the granularity of the PMR Elasticity Buffer Size field. Value: 0h – Bytes; 1h – 1 KiB; 2h – 1 MiB; 3h – 1 GiB; 4h to Fh – Reserved

3.1.3.26 Offset E10h: PMRSWTP – Persistent Memory Region Sustained Write Throughput

This optional property identifies to the host the maximum PMR sustained write throughput. A value of 0h in this property indicates to the host that no information regarding the PMR sustained write throughput is available.

Figure 67: Offset E10h: PMRSWTP – Persistent Memory Region Sustained Write Throughput

Bits	Type	Reset	Description
31:8	RO	Impl Spec	**PMR Sustained Write Throughput (PMRSWTV):** Indicates the sustained write throughput of the PMR at the maximum payload size specified by the applicable NVMe Transport binding specification (e.g. the PCIe TLP payload size, as specified in the Max_Payload_Size (MPS) field of the PCI Express Device Control (PXDC) register). The actual sustained write throughput of the PMR is equal to the value in this field multiplied by the units specified by the PMR Sustained Write Throughput Units field.
7:4	RO	0h	Reserved
3:0	RO	Impl Spec	**PMR Sustained Write Throughput Units (PMRSWTU):** Indicates the granularity of the PMR Sustained Write Throughput field. Value: 0h – Bytes per second; 1h – 1 KiB / s; 2h – 1 MiB / s; 3h – 1 GiB / s; 7h to Fh – Reserved

3.1.3.27 Offset E14h: PMRMSCL – Persistent Memory Region Memory Space Control Lower

This optional property and the PMRMSCU property specify how the controller references the Persistent Memory Region with host-supplied addresses. If the controller supports the Persistent Memory Region's controller memory space (PMRCAP.CMSS), this property is mandatory. Otherwise, this property is reserved. The host shall access this register with aligned 32-bit accesses.

This property shall not be reset by Controller Reset.

Figure 68: Offset E14h: PMRMSCL – Persistent Memory Region Memory Space Control Lower

Bits	Type	Reset	Description
31:12	RW	0h	**Controller Base Address (CBA):** This field specifies the 20 least significant bits of the 52 most significant bits of the 64-bit base address for the Persistent Memory Region's controller address range. The Persistent Memory Region's controller base address and its size determine its controller address range. The 64-bit base address specified by this field and PMRMSCU.CBA when the CMSE bit is set to '1' shall be valid only under the following conditions: a) no part of the Persistent Memory Region's controller address range is greater than $2^{64} - 1$; and b) if the Controller Memory Buffer's controller memory space is enabled, then the Persistent Memory Region's controller address range does not overlap the Controller Memory Buffer's controller address range.
11:02	RO	0h	Reserved
01	RW	0b	**Controller Memory Space Enable (CMSE):** This bit specifies whether addresses supplied by the host are permitted to reference the Persistent Memory Region. If this bit is set to '1' and the controller base address is valid, then the Persistent Memory Region's controller memory space is enabled. Otherwise, the controller memory space is not enabled. If the Persistent Memory Region's controller memory space is enabled, then addresses supplied by the host that fall within the Persistent Memory Region's controller address range shall reference the Persistent Memory Region. If the Persistent Memory Region's controller memory space is not enabled, then no address supplied by the host shall reference the Persistent Memory Region. Instead, such addresses shall reference memory spaces other than the Persistent Memory Region.
00	RO	0b	Reserved

3.1.3.28 Offset E18h: PMRMSCU – Persistent Memory Region Memory Space Control Upper

This optional property and the PMRMSCL property specify how the controller references the Persistent Memory Region with host-supplied addresses. If the controller supports the Persistent Memory Region's controller memory space (PMRCAP.CMSS), this property is mandatory. Otherwise, this property is reserved. The host shall access this register with aligned 32-bit accesses.

This register shall not be reset by Controller Reset.

Figure 69: Offset E18h: PMRMSCU – Persistent Memory Region Memory Space Control Upper

Bits	Type	Reset	Description
31:00	RW	0h	**Controller Base Address (CBA):** This field specifies the 32 most significant bits of the 52 most significant bits of the 64-bit base address for the Persistent Memory Region's controller address range. The Persistent Memory Region's controller base address and its size determine its controller address range.

3.2 NVM Subsystem Entities

3.2.1 Namespaces

3.2.1.1 Namespace Overview

A namespace is a formatted quantity of non-volatile memory that may be directly accessed by a host. A namespace ID (NSID) is an identifier used by a controller to provide access to a namespace.

3.2.1.2 Valid and Invalid NSIDs

Valid NSIDs are the range of possible NSIDs that may be used to refer to namespaces that exist in the NVM subsystem. Any NSID is valid, except if that NSID is 0h or greater than the Number of Namespaces

field reported in the Identify Controller data structure (refer to Figure 275). NSID FFFFFFFFh is a broadcast value that is used to specify all namespaces. An invalid NSID is any value that is not a valid NSID and is also not the broadcast value.

Valid NSIDs are:

a) allocated or unallocated in the NVM subsystem; and
b) active or inactive for a specific controller.

3.2.1.3 Allocated and Unallocated NSID Types

In the NVM subsystem, a valid NSID is:

a) an allocated NSID; or
b) an unallocated NSID.

Allocated NSIDs refer to namespaces that exist in the NVM subsystem. Unallocated NSIDs do not refer to any namespaces that exist in the NVM subsystem.

3.2.1.4 Active and Inactive NSID Types

For a specific controller, an allocated NSID is:

a) an active NSID; or
b) an inactive NSID.

Active NSIDs for a controller refer to namespaces that are attached to that controller. Allocated NSIDs that are inactive for a controller refer to namespaces that are not attached to that controller.

Unallocated NSIDs are inactive NSIDs for all controllers in the NVM subsystem.

An allocated NSID may be an active NSID for some controllers and an inactive NSID for other controllers in the same NVM subsystem if the namespace that the NSID refers to is attached to some controllers, but not all controllers, in the NVM subsystem.

Refer to section 8.11 for actions associated with a namespace being detached or deleted.

3.2.1.5 NSID and Namespace Relationships

Unless otherwise noted, specifying an inactive NSID in a command that uses the Namespace Identifier (NSID) field shall cause the controller to abort the command with a status code of Invalid Field in Command. Specifying an invalid NSID in a command that uses the NSID field shall cause the controller to abort the command with a status code of Invalid Namespace or Format.

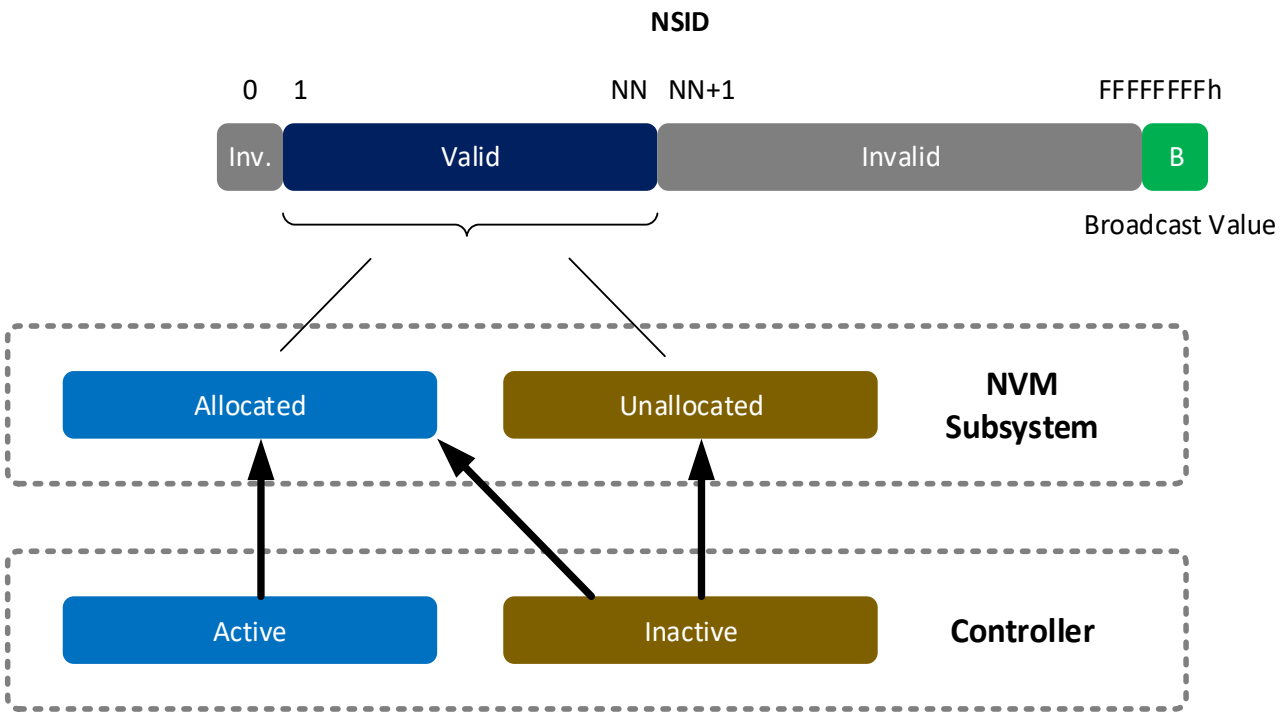
The following table summarizes the valid NSID types and Figure 70 visually shows the NSID types and how they relate.

Figure 70: NSID Types and Relationship to Namespace

Valid NSID Type	NSID relationship to namespace	Reference
Unallocated	Does not refer to any namespace that exists in the NVM subsystem	3.2.1.3
Allocated	Refers to a namespace that exists in the NVM subsystem	3.2.1.3
Inactive	Does not refer to a namespace that is attached to the controller[1]	3.2.1.4
Active	Refers to a namespace that is attached to the controller	3.2.1.4

Notes:
1. If allocated, refers to a namespace that is not attached to the controller. If unallocated, does not refer to any namespace.

Figure 71: NSID Types



3.2.1.6 NSID and Namespace Usage

If Namespace Management (refer to section 8.11), ANA Reporting (refer to section 8.1), or NVM Sets (refer to section 3.2) capabilities are supported, then NSIDs shall be unique within the NVM subsystem (e.g., NSID of 3 shall refer to the same physical namespace regardless of the accessing controller). If the Namespace Management, ANA Reporting, and NVM Sets capabilities are not supported, then NSIDs:

a) for shared namespaces shall be unique within the NVM subsystem; and
b) for private namespaces are not required to be unique within the NVM subsystem.

The Identify command (refer to section 5.17) may be used to determine the active NSIDs for a controller and the allocated NSIDs in the NVM subsystem.

If the MNAN field (refer to Figure 275) is cleared to 0h, then the maximum number of allocated NSIDs is the same as the value reported in the NN field (refer to Figure 275). If the MNAN field is non-zero, then the maximum number of allocated NSIDs may be less than the number of namespaces (e.g., an NVM subsystem may support a maximum valid NSID value (i.e., the NN field) set to 1,000,000 but support a maximum of 10 allocated NSID values).

To determine the active NSIDs for a particular controller, the host may follow either of the following methods:

1. Issue an Identify command with the CNS field cleared to 0h for each valid NSID (based on the Number of Namespaces value (i.e., MNAM field or NN field) in the Identify Controller data structure). If a non-zero data structure is returned for a particular NSID, then that is an active NSID; or
2. Issue an Identify command with a CNS field set to 2h to retrieve a list of up to 1,024 active NSIDs. If there are more than 1,024 active NSIDs, continue to issue Identify commands with a CNS field set to 2h until all active NSIDs are retrieved.

To determine the allocated NSIDs in the NVM subsystem, the host may issue an Identify command with the CNS field set to 10h to retrieve a list of up to 1,024 allocated NSIDs. If there are more than 1,024 allocated NSIDs, continue to issue Identify commands with a CNS field set to 10h until all allocated NSIDs are retrieved.

Namespace IDs may change across power off conditions. However, it is recommended that Namespace IDs remain static across power off conditions in order to avoid issues with host software. To determine if the same namespace has been encountered, the host may use the:

a) UUID field in the Namespace Identification Descriptor (refer to Figure 277), if present;
b) NGUID field in the Identify Namespace data (refer to the applicable I/O Command Set specification) or in the Namespace Identification Descriptor, if present; or
c) EUI64 field in the Identify Namespace data or in the Namespace Identification Descriptor, if present.

UIDREUSE bit in the NSFEAT field (refer to Figure 280 or the Identify Namespace data structure in the NVM Command Set Specification, if applicable) indicates NGUID and EUI64 reuse characteristics.

If Asymmetric Namespace Access Reporting is supported (i.e., bit 3 set to '1' in the CMIC field in the Identify Controller data structure (refer to Figure 275)), refer to the applicable I/O Command Set specification for additional detail, if any.

A namespace may or may not have a relationship to a Submission Queue; this relationship is determined by the host software implementation. The controller shall support access to any active namespace from any I/O Submission Queue.

3.2.1.7 I/O Command Set Associations

A namespace is associated with exactly one I/O Command Set. For I/O commands and I/O Command Set specific Admin commands, the I/O Command Set with which a submission queue entry is associated is determined by the Namespace Identifier (NSID) field in the command.

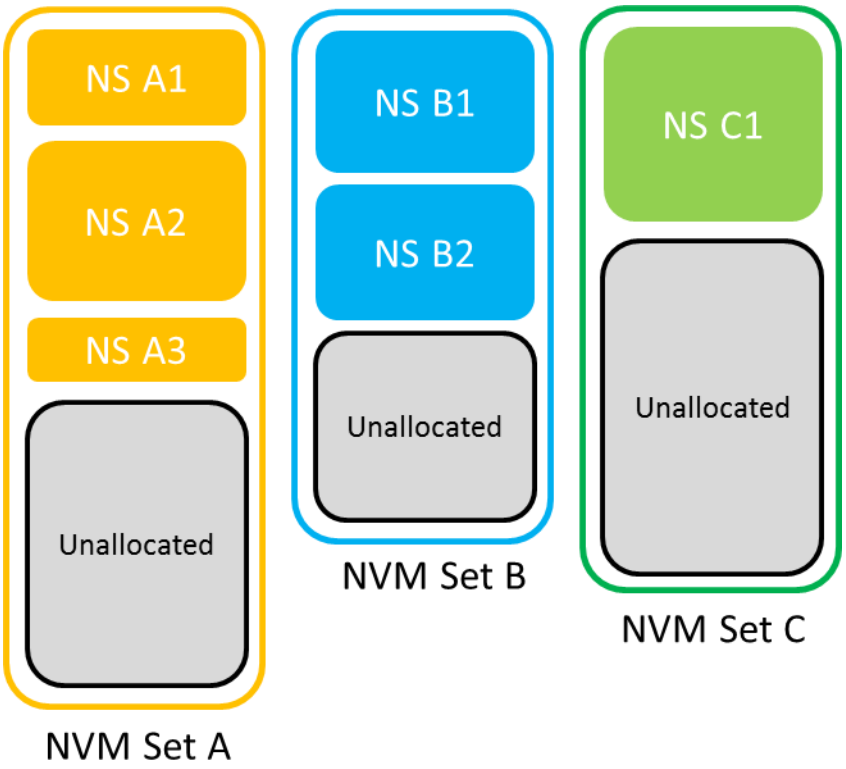
An NVM subsystem may contain namespaces each of which is associated with a different I/O Command Set. A controller may support attached namespaces that use any of the I/O Command Sets that the controller simultaneously supports as indicated in the I/O Command Set Profile (refer to section 5.27.1.21).

3.2.2 NVM Sets

An NVM Set is a collection of NVM that is separate (logically and potentially physically) from NVM in other NVM Sets. One or more namespaces may be created within an NVM Set and those namespaces inherit the attributes of the NVM Set. A namespace is wholly contained within a single NVM Set and shall not span more than one NVM Set.

Figure 72 shows an example of three NVM Sets. NVM Set A contains three namespaces (NS A1, NS A2, and NS A3). NVM Set B contains two namespaces (NS B1 and NS B2). NVM Set C contains one namespace (NS C1). Each NVM Set shown also contains 'Unallocated' regions that consist of NVM that is not yet allocated to a namespace.

Figure 72: NVM Sets and Associated Namespaces



There is a subset of Admin commands that are NVM Set aware as described in Figure 73.

Figure 73: NVM Set Aware Admin Commands

Admin Command	Details
Identify	• The Identify Namespace data structure includes the associated NVM Set Identifier. • The NVM Set List data structure includes attributes for each NVM Set.
Capacity Management	• The Create NVM Set action returns the NVM Set Identifier of the NVM Set that is created. • The Delete NVM Set action includes the NVM Set Identifier of the NVM Set that is to be deleted.
Namespace Management	• The create action includes the NVM Set Identifier as a host specified field.
Get Features and Set Features	• The Read Recovery Level Feature specifies the associated NVM Set Identifier. • The Predictable Latency Mode Config Feature specifies the associated NVM Set Identifier. • The Predictable Latency Mode Window Feature specifies the associated NVM Set Identifier.
Create I/O Submission Queue	• The Create I/O Submission Queue command includes the associated NVM Set Identifier.

The host determines the NVM Sets present and their attributes using the Identify command with CNS value of 04h to retrieve the NVM Set List (refer to Figure 278). For each NVM Set, the attributes include:

- an identifier associated with the NVM Set;
- the optimal size for writes to the NVM Set;
- the total capacity of the NVM Set; and
- the unallocated capacity for the NVM Set.

An NVM Set Identifier is a 16-bit value that specifies the NVM Set with which an action is associated. An NVM Set Identifier is unique with the NVM subsystem. An NVM Set Identifier may be specified in NVM Set aware Admin commands (refer to Figure 73). An NVM Set Identifier value of 0h is reserved and is not a valid NVM Set Identifier. Unless otherwise specified, if the host specifies an NVM Set Identifier cleared to 0h for a command that requires an NVM Set Identifier, then that command shall abort with a status code of Invalid Field in Command.

Each NVM Set is associated with exactly one Endurance Group (refer to section 3.2.3).

The NVM Set with which a namespace is associated is reported in the Identify Namespace data structure (refer to the applicable NVMe I/O Command Set specification). When a host creates a namespace using the Namespace Management command, the host specifies the NVM Set Identifier of the NVM Set that the namespace is to be created in. The namespace that is created inherits attributes from the NVM Set (e.g., the optimal write size to the NVM).

If NVM Sets are supported, then all controllers in the NVM subsystem shall:

- Indicate support for NVM Sets in the Controller Attributes field in the Identify Controller data structure;
- Support the NVM Set Identifier in all commands that use the NVM Set Identifier;
- Support the NVM Set List for the Identify command;
- Indicate the NVM Set Identifier with which the namespace is associated in the Identify Namespace data structure;
- Support Endurance Groups; and
- For each NVM Set, indicate the associated Endurance Group as an attribute.

If support for NVM Sets is not reported (i.e., the NVM Sets bit is cleared to ‘0’ in the CTRATT field; refer to Figure 275), then the NVM Set Identifier field shall be cleared to 0h in all commands and data structures that support an NVM Set Identifier field.

3.2.3 Endurance Groups

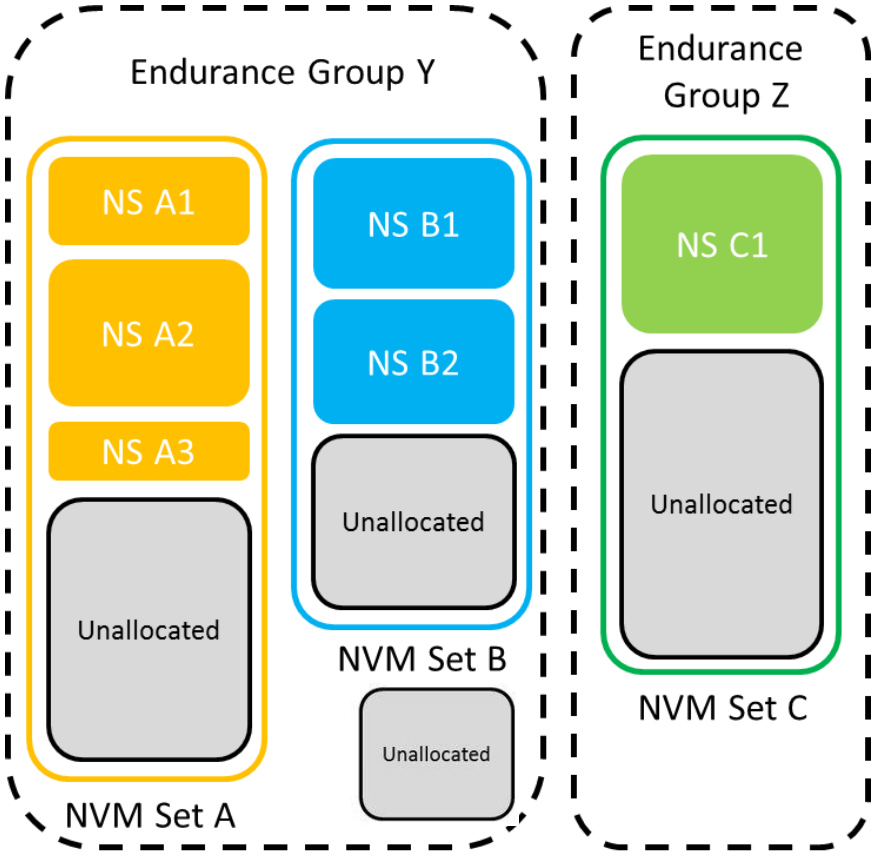
Endurance may be managed within a single NVM Set (refer to section 3.2.2) or across a collection of NVM Sets. Each NVM Set is associated with an Endurance Group (refer to Figure 278). If two or more NVM Sets have the same Endurance Group Identifier, then endurance is managed by the NVM subsystem across that collection of NVM Sets. If only one NVM Set is associated with a specific Endurance Group Identifier, then endurance is managed locally to that NVM Set. An Endurance Group shall be part of only one domain (refer to section 3.2.4).

An Endurance Group Identifier is a 16-bit value that specifies the Endurance Group with which an action is associated. An Endurance Group Identifier is unique within the NVM subsystem. An Endurance Group Identifier value of 0h is reserved and is not a valid Endurance Group Identifier. Unless otherwise specified, if the host specifies an Endurance Group Identifier cleared to 0h for a command that requires an Endurance Group Identifier, then that command shall abort with a status code of Invalid Field in Command.

The information that describes an Endurance Group is indicated in the Endurance Group Information log page (refer to section 5.16.1.10).

Figure 74 shows Endurance Groups added to the example in Figure 72. In this example, the endurance of NVM Set A and NVM Set B are managed together as part of Endurance Group Y, while the endurance of NVM Set C is managed only within NVM Set C which is the only NVM Set that is part of Endurance Group Z.

Figure 74: NVM Sets and Associated Namespaces



If Endurance Groups are supported, then the NVM subsystem and all controllers shall:

- indicate support for Endurance Groups in the Controller Attributes field in the Identify Controller data structure;
- indicate the Endurance Group Identifier with which the namespace is associated in the Identify Namespace data structure;
- support the Endurance Group Information log page; and
- support the Endurance Group Event Aggregate log page.

If Endurance Groups are not supported and the host sends a command in which an Endurance Group Identifier field is defined (e.g., Get Log Page), then that field shall be ignored by the controller.

If Endurance Groups are not supported and the controller returns information to the host that contains an Endurance Group Identifier field, then that field shall be cleared to 0h.

3.2.3.1 Configuring and Managing Endurance Group Events

The host may configure asynchronous events to be triggered when certain events occur for an Endurance Group. The host submits a Set Features command specifying the Endurance Group Event Configuration feature (refer to section 5.27.1.20), the Endurance Group, and the specific event(s) that shall trigger adding an entry to the Endurance Group Event Aggregate log page (refer to section 5.16.1.15).

The host configures events using a Set Features command for each Endurance Group.

The host submits a Set Features command specifying the Asynchronous Event Configuration feature (refer to section 5.27.1.8) with the Endurance Group Event Aggregate Log Change Notices bit set to ‘1’ to specify that adding an entry to the Endurance Group Event Aggregate log page shall trigger an Endurance Group Event Aggregate Log Page Change Notice event to the host (refer to Figure 353).

The host determines the Endurance Groups that have outstanding events by reading the Endurance Group Event Aggregate log page. An entry is returned for each Endurance Group that has an event outstanding. The host may use the Endurance Group Identifier Maximum value reported in the Identify Controller data structure to determine the maximum size of this log page.

To determine the specific event(s) that have occurred for a reported Endurance Group, the host reads the Endurance Group Information log page (refer to Figure 217) for that Endurance Group. The Critical Warning field indicates the event(s) that have occurred (e.g., that all namespaces in the Endurance Group have been placed in read-only mode). All events for an Endurance Group are cleared if the controller successfully processes a read for the Endurance Group Information log page for that Endurance Group, where the Get Log Page command has the Retain Asynchronous Event bit cleared to '0'. If the Critical Warning field in the Endurance Group Information log page is cleared to 0h, then events for that Endurance Group are not reported in the Endurance Group Event Aggregate log page.

3.2.4 Domains and Divisions

3.2.4.1 Overview

An NVM subsystem may be made up of a single domain or multiple domains (i.e., two or more). A domain is the smallest indivisible unit that shares state (e.g., power state, capacity information). An NVM subsystem that supports multiple domains shall support Asymmetric Namespace Access Reporting (refer to section 8.1).

A common example of a simple implementation of an NVM subsystem is one that consists of a single domain (i.e., multiple domains are not supported).

Each domain is independent, and the boundaries between domains are communication boundaries (e.g., fault boundaries, management boundaries). If multiple domains are present in an NVM subsystem, then those domains cooperate in the operation of that NVM subsystem. If a domain is unable to cooperate in the operation of the NVM subsystem, then the NVM subsystem has become divided.

A division is an event (e.g., failure of a domain) or action (e.g. management action or reconfiguration) within the NVM subsystem that affects communication between the domains contained in the NVM subsystem (refer to Figure 75 and Figure 76). If a division exists, global state within the NVM subsystem may be impacted (e.g., a controller may only have information about the state of the domains with which the controller is able to communicate). A division event or action may:

- affect access to namespaces (refer to section 8.1); or
- impact operations that have NVM subsystem scope (e.g., TNVMCAP, sanitize, format, SMART information).

A domain is comprised of:

- zero or more controllers; and
- zero or more NVM Endurance Groups.

If an NVM subsystem supports multiple domains, then all controllers in that NVM subsystem shall:

- set the MDS bit to '1' in the CTRATT field in the Identify Controller data structure (refer to Figure 275);
- set the Domain Identifier in each Endurance Group descriptor, if supported, to a non-zero value; and
- set the Domain Identifier in each Identify Controller data structure to a non-zero value.

If an NVM subsystem supports multiple domains, then controllers in that NVM subsystem may:

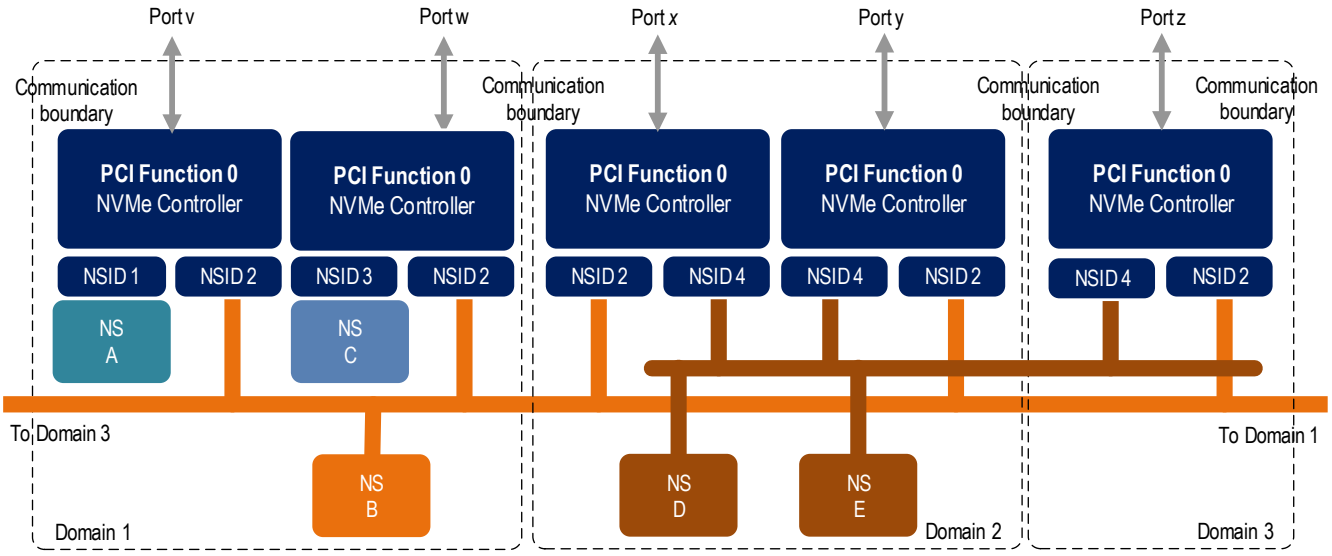
- support Endurance Groups (refer to Endurance Groups bit in the CTRATT field of Identify Controller data structure).

For an NVM subsystem that supports multiple domains, each domain shall be assigned a domain identifier that is unique within the NVM subsystem (refer to the Domain Identifier field in Figure 275 and section

3.2.4.3). For an NVM subsystem that does not support multiple domains, Domain Identifier fields are cleared to 0h.

Figure 75 shows an example of an NVM subsystem that consists of three domains. Domain 1 contains two controllers and some amount of NVM storage capacity which has been allocated to two private namespaces (i.e., NS A and NS C) and a shared namespace (i.e., NS B). Domain 2 contains two controllers and some amount of NVM storage capacity which has been allocated to two shared namespaces (i.e., NS D and NS E). Domain 3 contains one controller, and no NVM storage capacity.

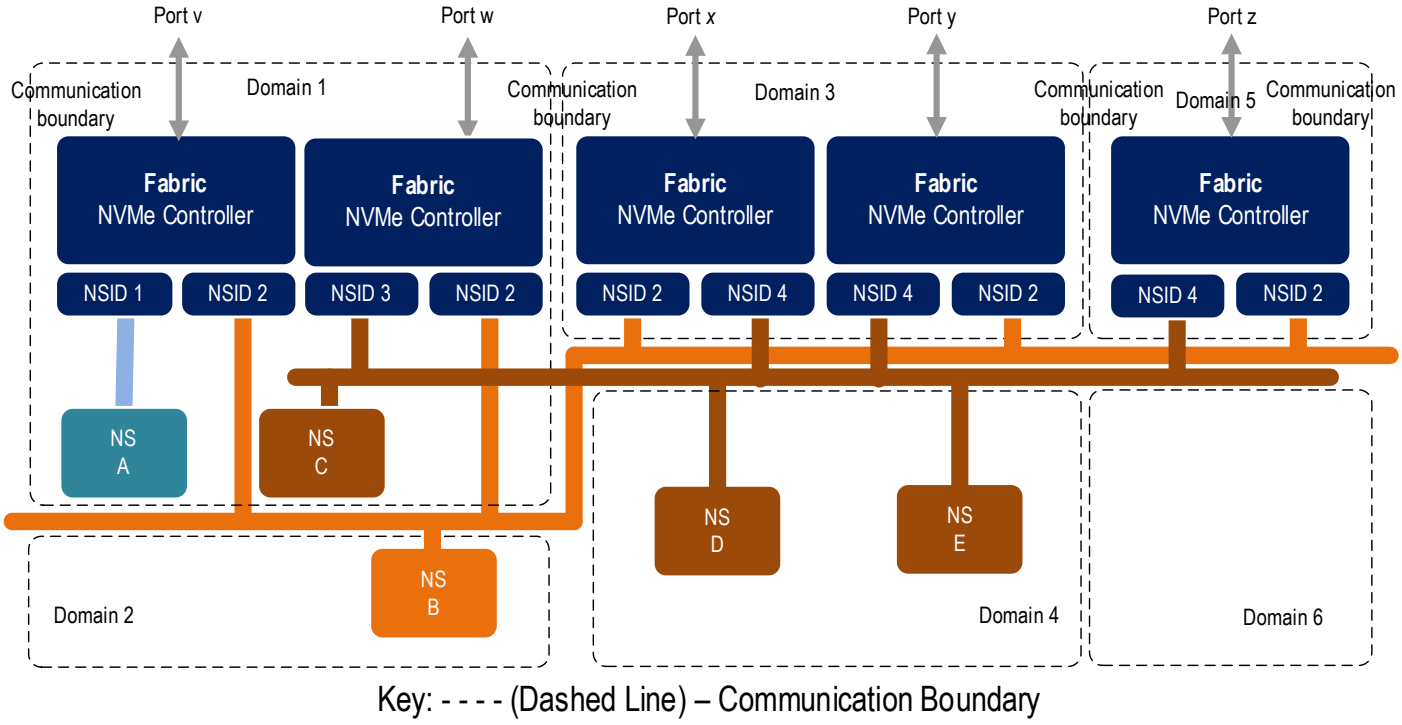
Figure 75: Example 1 Domain Structure



If, in the example shown in Figure 75, a division event occurs that results in Domain 1 no longer being able to communicate with Domain 2 and Domain 3, then the NVM subsystem would consist of two parts. The first part consists of Domain 1 and the second part consists of Domain 2 and Domain 3.

Figure 76 shows an example of an NVM subsystem that consists of six domains, of which, three are domains that contain controllers. Domain 1 is a domain that contains two controllers and some amount of NVM storage capacity from which NVM Endurance Groups have been created that contain a private namespace (i.e., NS A) and a shared namespace (i.e., NS C). Domain 2 is a domain that contains no controllers and contains some amount of NVM storage capacity from which NVM Endurance Groups have been created that contain a shared namespace (i.e., NS B). Domain 3 is a domain that contains two controllers and no NVM storage capacity. Domain 4 is a domain that contains no controllers and contains some amount of NVM storage capacity from which NVM Endurance Groups have been created that contain two shared namespaces (i.e., NS D and NS E). Domain 5 is a domain that contains one controller and no NVM storage capacity. Domain 6 is a domain that contains no controllers and no NVM storage capacity allocated to an NVM Endurance Group (i.e., an empty domain).

Figure 76: Example 2 Domain Structure



3.2.4.2 Domains and Reservations

If an NVM subsystem supports multiple domains and Persistent Reservations (refer to section 8.19), then resumption after a division event (e.g., resumption of operation, resumption of communication) requires that all persistent reservation state within the domains in the NVM subsystem that are no longer divided be synchronized (i.e., updated).

If the reservation state for a namespace is not synchronized, then the ANA Group that contains that namespace shall transition to the ANA Inaccessible state (refer to section 8.1.3.3) and remain in that state until the Persistent Reservation state is synchronized. If the Persistent Reservation state is not able to be synchronized, then:

- a transition to the ANA Persistent Loss state occurs and commands are processed as described in section 8.1.3.4; or
- the controller may stop processing commands and set the Controller Fatal Status (CSTS.CFS) bit to ‘1’ (refer to section 9.5).

3.2.4.3 Domain Identifier Use (Informative)

Domain Identifier values indicate the parts of the NVM subsystem that comprise a domain.

The host may use these values to determine which Endurance Groups (refer to section 3.2.3) are contained in the same domain and which are contained in a different domain. Examples of host use of the domain identifier include:

- host data redundancy software (e.g., RAID) that may use the Endurance Group’s Domain Identifier to determine which Endurance Groups may fail together (e.g., Endurance Groups in the same domain) and which Endurance Groups may fail independently (e.g., Endurance Groups in different domains); and
- host application software may use the controller’s Domain Identifier to determine which controllers share domains (e.g., controllers that may fail together) and which controllers are a part of different domains (e.g., controllers that may fail independently).

3.3 NVM Queue Models

The NVM Express interface is based on a paired Submission and Completion Queue mechanism. Commands are placed by host software into a Submission Queue. Completions are placed into the associated Completion Queue by the controller. When using a memory-based transport queue model, multiple Submission Queues may utilize the same Completion Queue. When using a message-based transport queue model each Submission Queue maps to a single Completion Queue.

3.3.1 Memory-based Transport Queue Model

3.3.1.1 Queue Setup and Initialization

To setup and initialize I/O Submission Queues and I/O Completion Queues for use, host software follows these steps:

1. Configures the Admin Submission and Completion Queues by initializing the Admin Queue Attributes (AQA), Admin Submission Queue Base Address (ASQ), and Admin Completion Queue Base Address (ACQ) properties appropriately;
2. Configures the size of the I/O Submission Queues (CC.IOSQES) and I/O Completion Queues (CC.IOCQES);
3. Submits a Set Features command with the Number of Queues attribute set to the requested number of I/O Submission Queues and I/O Completion Queues. The completion queue entry for this Set Features command indicates the number of I/O Submission Queues and I/O Completion Queues allocated by the controller;
4. Determines the maximum number of entries supported per queue (CAP.MQES) and whether the queues are required to be physically contiguous (CAP.CQR);
5. Creates I/O Completion Queues within the limitations of the number allocated by the controller and the queue attributes supported (maximum entries and physically contiguous requirements) by using the Create I/O Completion Queue command; and
6. Creates I/O Submission Queues within the limitations of the number allocated by the controller and the queue attributes supported (maximum entries and physically contiguous requirements) by using the Create I/O Submission Queue command.

At the end of this process, I/O Submission Queues and I/O Completion Queues have been setup and initialized and may be used to complete I/O commands.

3.3.1.2 Queue Usage

The submitter of entries to a memory-based transport queue uses the current Tail entry pointer to identify the next open queue slot. The submitter increments the Tail entry pointer after placing the new entry to the open queue slot. If the Tail entry pointer increment exceeds the queue size, the Tail entry shall roll to zero. The submitter may continue to place entries in free queue slots as long as the Full queue condition is not met (refer to section 3.3.1.5).

Note: The submitter shall take queue wrap conditions into account.

The consumer of entries on a memory-based transport queue uses the current Head entry pointer to identify the slot containing the next entry to be consumed. The consumer increments the Head entry pointer after consuming the next entry from the queue. If the Head entry pointer increment exceeds the queue size, the Head entry pointer shall roll to zero. The consumer may continue to consume entries from the queue as long as the Empty queue condition is not met (refer to section 3.3.1.3).

Note: The consumer shall take queue wrap conditions into account.

Creation and deletion of memory-based transport Submission Queue and associated Completion Queues are required to be ordered correctly by host software. Host software creates the Completion Queue before creating any associated Submission Queue. Submission Queues may be created at any time after the associated Completion Queue is created. Host software deletes all associated Submission Queues prior to deleting a Completion Queue. To abort all commands submitted to the Submission Queue host software issues a Delete I/O Submission Queue command for that queue (refer to section 3.3.1.3).

Host software writes the Submission Queue Tail Doorbell and the Completion Queue Head Doorbell (refer to the Transport Specific Controller Properties section in the NVMe over PCIe Transport Specification) to communicate new values of the corresponding entry pointers to the controller. If host software writes an invalid value to the Submission Queue Tail Doorbell or Completion Queue Head Doorbell property and an Asynchronous Event Request command is outstanding, then an asynchronous event is posted to the Admin Completion Queue with a status code of Invalid Doorbell Write Value. The associated queue is then deleted and recreated by host software. For a Submission Queue that experiences this error, the controller may complete previously consumed commands; no additional commands are consumed. This condition may be caused by host software attempting to add an entry to a full Submission Queue or remove an entry from an empty Completion Queue.

Host software checks completion queue entry Phase Tag (P) bits in memory to determine whether new completion queue entries have been posted (refer to section 3.3.3.2.2). The Completion Queue Tail pointer is only used internally by the controller and is not visible to the host. The controller uses the SQ Head Pointer (SQHD) field in completion queue entries to communicate new values of the Submission Queue Head Pointer to the host. A new SQHD value indicates that submission queue entries have been consumed, but does not indicate either execution or completion of any command. Refer to section 3.3.3.2.

A submission queue entry is submitted to the controller when the host writes the associated Submission Queue Tail Doorbell with a new value that indicates that the Submission Queue Tail Pointer has moved to or past the slot in which that submission queue entry was placed. A Submission Queue Tail Doorbell write may indicate that one or more submission queue entries have been submitted.

A submission queue entry has been consumed by the controller when a completion queue entry is posted that indicates that the Submission Queue Head Pointer has moved past the slot in which that submission queue entry was placed. A completion queue entry may indicate that one or more submission queue entries have been consumed.

A completion queue entry is posted to the Completion Queue when the controller write of that completion queue entry to the next free Completion Queue slot inverts the Phase Tag (P) bit from its previous value in memory (refer to section 3.3.3.2.2). The controller may generate an interrupt to the host to indicate that one or more completion queue entries have been posted.

A completion queue entry has been consumed by the host when the host writes the associated Completion Queue Head Doorbell with a new value that indicates that the Completion Queue Head Pointer has moved past the slot in which that completion queue entry was placed. A Completion Queue Head Doorbell write may indicate that one or more completion queue entries have been consumed.

Once a submission queue entry or a completion queue entry has been consumed, the slot in which it was placed is free and available for reuse. Altering a submission queue entry after that entry has been submitted but before that entry has been consumed results in undefined behavior. Altering a completion queue entry after that entry has been posted but before that entry has been consumed results in undefined behavior.

3.3.1.2.1 Completion Queue Flow Control

If there are no free slots in a Completion Queue, then the controller shall not post status to that Completion Queue until slots become available. In this case, the controller may stop processing additional submission queue entries associated with the affected Completion Queue until slots become available. The controller shall continue processing for other Submission Queues not associated with the affected Completion Queue.

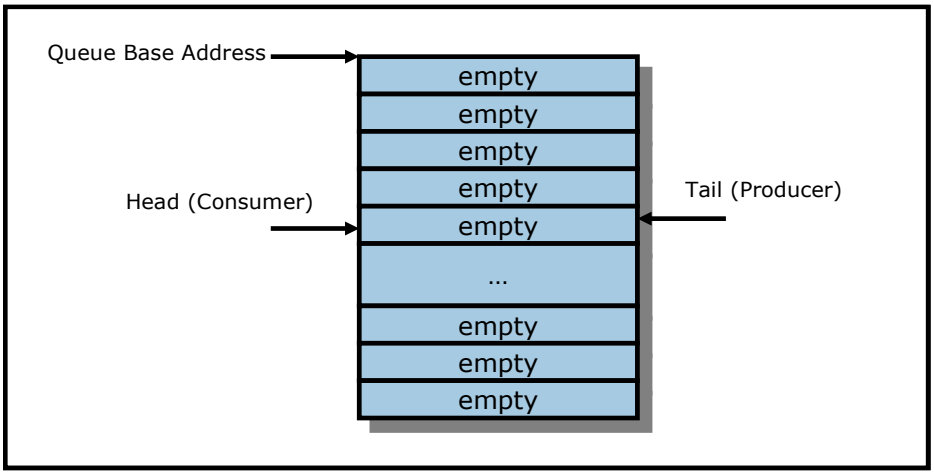
3.3.1.3 Queue Abort

To abort a large number of commands, the recommended procedure is to delete and recreate the I/O Submission Queue. Specifically, to abort all commands that are submitted to the I/O Submission Queue host software should issue a Delete I/O Submission Queue command for that queue. After the queue has been successfully deleted, indicating that all commands have been completed or aborted, then host software should recreate the queue by submitting a Create I/O Submission Queue command. Host software may then re-submit commands to the associated I/O Submission Queue.

3.3.1.4 Empty Queue

The queue is Empty when the Head entry pointer equals the Tail entry pointer. Figure 77 defines the Empty Queue condition.

Figure 77: Empty Queue Definition

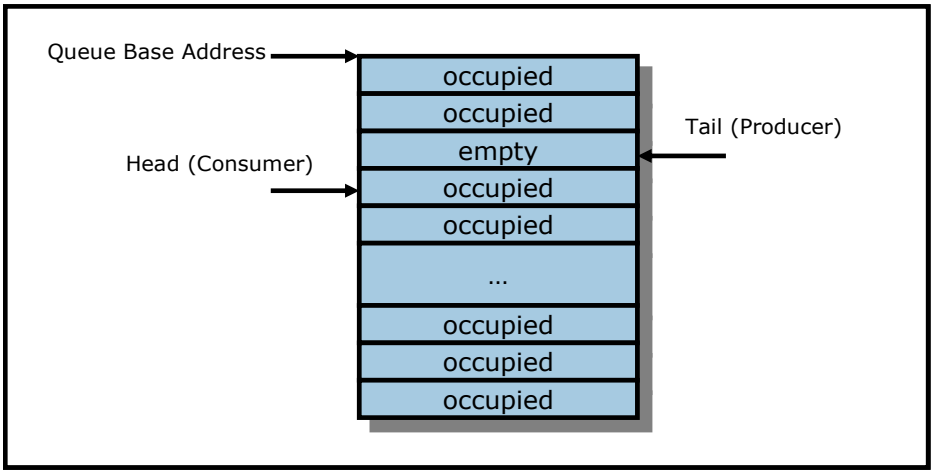


3.3.1.5 Full Queue

The queue is Full when the Head equals one more than the Tail. The number of entries in a queue when full is one less than the queue size. Figure 78 defines the Full Queue condition.

Note: Queue wrap conditions shall be taken into account when determining whether a queue is Full.

Figure 78: Full Queue Definition



3.3.2 Message-based Transport Queue Model

For NVMe over Fabrics, a queue is a unidirectional communication channel that is used to send capsules between a host and a controller. A host uses Submission Queues to send command capsules (refer to section 3.3.2.1.1) to a controller. A controller uses Completion Queues to send response capsules (refer to section 3.3.2.1.2) to a host. Submission and Completion Queues are created in pairs using the Connect command (refer to section 3.3.2.2).

The NVMe Transport is responsible for delivering command capsules to the controller and notifying the controller of capsule arrival in a transport-specific fashion.

Altering a command capsule between host submission to the Submission Queue and transport delivery of that capsule to the controller results in undefined behavior.

NVMe Transports are not required to provide any additional end-to-end flow control. Specific NVMe Transports may require low level flow control for congestion avoidance and reliability; any such additional NVMe Transport flow control is outside the scope of this specification.

Flow control differs for Submission Queues (refer to section 3.3.2.1.1, section 3.3.2.6, and section 3.3.2.7) and Completion Queues (refer to sections 3.3.2.1.2, section 3.3.2.8, and section 3.3.1.2.1).

3.3.2.1 Capsules and Data Transfers

This section describes capsules and data transfer mechanisms necessary to support message-based transport queues. These mechanisms are used for Fabrics commands, Admin commands, and I/O commands when using the message-based transport queue model.

A capsule is an NVMe unit of information exchanged between a host and a controller. A capsule may contain commands, responses, SGLs, and/or data. The data may include user data (e.g., logical block data and metadata that is transferred as a contiguous part of the logical block) and data structures associated with the command.

The capsule size for the Admin Queue commands and responses is fixed and defined in the NVMe Transport binding specification. The controller indicates in the Identify Controller data structure the capsule command and response sizes that the host shall use with I/O commands.

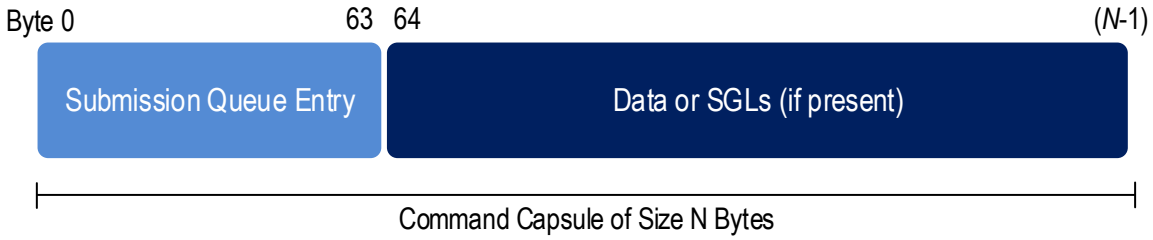
The controller shall support SGL based data transfers for commands on both the Admin Queue and I/O Queues. Data may be transferred within the capsule or through memory transactions based on the underlying NVMe Transport as indicated in the SGL descriptors associated with the command capsule. The SGL types supported by an NVMe Transport are specified in the NVMe Transport binding specification.

The value of unused and not reserved capsule fields (e.g., the capsule is larger than the command / response and associated data) is undefined and shall not be interpreted by the recipient.

3.3.2.1.1 Command Capsules

A command capsule is sent from a host to a controller. It contains a submission queue entry (SQE) and may optionally contain data or SGLs. The SQE is 64 bytes in size and contains the Admin command, I/O command, or Fabrics command to be executed.

Figure 79: Command Capsule



The Command Identifier field in the SQE shall be unique among all outstanding commands associated with that queue. If there is data or additional SGLs to be transferred within the capsule, then the SGL descriptor in the SQE contains a Data Block, Segment Descriptor, or Last Segment Descriptor specifying an appropriate Offset address. The definition for the submission queue entry when the command is a Fabrics command is defined in Figure 80. The definition for the submission queue entry when the command is an Admin or I/O command is defined in section 3.3.3.1, where the Metadata Pointer field is reserved.

Figure 80: Fabrics Command Capsule – Submission Queue Entry Format

Bytes	Description
00	**Opcode (OPC):** Set to 7Fh to indicate a Fabrics command.

Figure 80: Fabrics Command Capsule – Submission Queue Entry Format

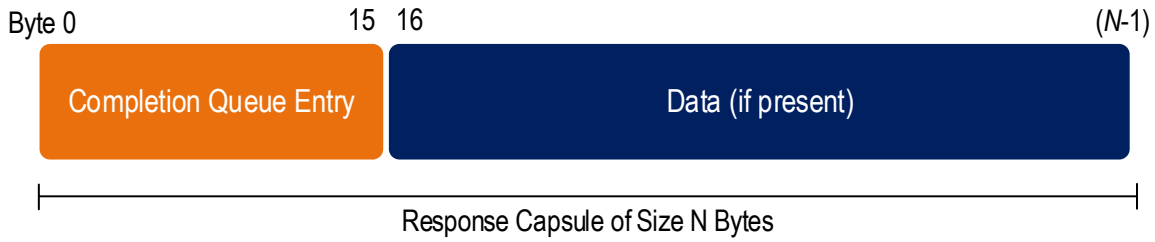
Bytes	Description
01	**PRP or SGL for Data Transfer (PSDT):** This field is described in Figure 86. Bits 7:6 of this field should be set to 01b, and may be cleared to 00b.
03:02	**Command Identifier (CID):** This field specifies a unique identifier for the command. The identifier shall be unique among all outstanding commands associated with a particular queue.
04	**Fabrics Command Type (FCTYPE):** This field specifies the Fabrics command transferred in the capsule. The Fabrics command types are defined in Figure 375. If this field is set to a reserved value, the command should be aborted with a status code of Invalid Field in Command.
39:05	Reserved
63:40	**Fabrics Command Type Specific:** This field is Fabrics command type specific.

3.3.2.1.2 Response Capsules

A response capsule is sent from the NVM subsystem to the host. It contains a completion queue entry (CQE) and may optionally contain data. The CQE is the completion queue entry associated with a previously issued command capsule.

If a command requests data and the SGL in the associated command capsule specifies a Data Block descriptor with an Offset, the data is included in the response capsule. If the SGL(s) in the command capsule specify a region in host memory, then data is transferred via memory transactions.

Figure 81: Response Capsule



The completion queue entry is 16 bytes in size and contains a two byte status field.

The definition for the completion queue entry for a Fabrics command is defined in Figure 82. The definition for the completion queue entry when the command is an Admin or I/O command is defined in section 3.3.3.2, where the SQ Identifier and Phase Tag fields are reserved because they are not used in NVMe over Fabrics. Use of the SQHD field depends on whether SQ flow control is disabled for the queue pair, refer to section 6.3.

Figure 82: Fabrics Response Capsule – Completion Queue Entry Format

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>07:00</td><td>The definition of this field is Fabrics response type specific.</td></tr>
<tr><td>09:08</td><td>SQ Head Pointer (SQHD): Indicates the current Submission Queue Head pointer for the associated Submission Queue. This field is reserved if SQ flow control is disabled for the queue pair (refer to section 6.3).</td></tr>
<tr><td>11:10</td><td>Reserved</td></tr>
<tr><td>13:12</td><td>Command Identifier (CID): Indicates the identifier of the command that is being completed.</td></tr>
<tr><td>15:14</td><td>Status (STS): Specifies status for the associated Fabrics command.
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>15:01</td><td>Status field as defined in section 3.3.3.2.1.</td></tr>
<tr><td>00</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>

3.3.2.1.3 Data Transfers

Data may be transferred within capsules or by memory transfers. SGLs are used to specify the location of data. Metadata, if transferred, is a contiguous part of the user data with which that metadata is associated. The SGL descriptor(s) (refer to section 4.1.2) specify whether the command's data is transferred through memory or within the capsule. The capsule may contain either SGLs or data (not a mixture of both) following the SQE. If additional SGLs are required, then the SGLs are included in the capsule immediately after the SQE. If an invalid offset is specified in an SGL descriptor, then a status code of SGL Offset Invalid shall be returned.

SGLs shall be supported within a capsule. The NVMe Transport binding specification defines the SGL Descriptor Types and Sub Types that are supported for the corresponding NVMe Transport. The NVMe Transport binding specification also specifies if SGLs may be supported in host memory.

3.3.2.1.3.1 Data and SGL Locations within a Command Capsule

The submission queue entry within the command capsule includes one SGL entry. If there are additional SGL entries to be transferred in the command capsule, then those entries shall be contiguous and located immediately after the submission queue entry.

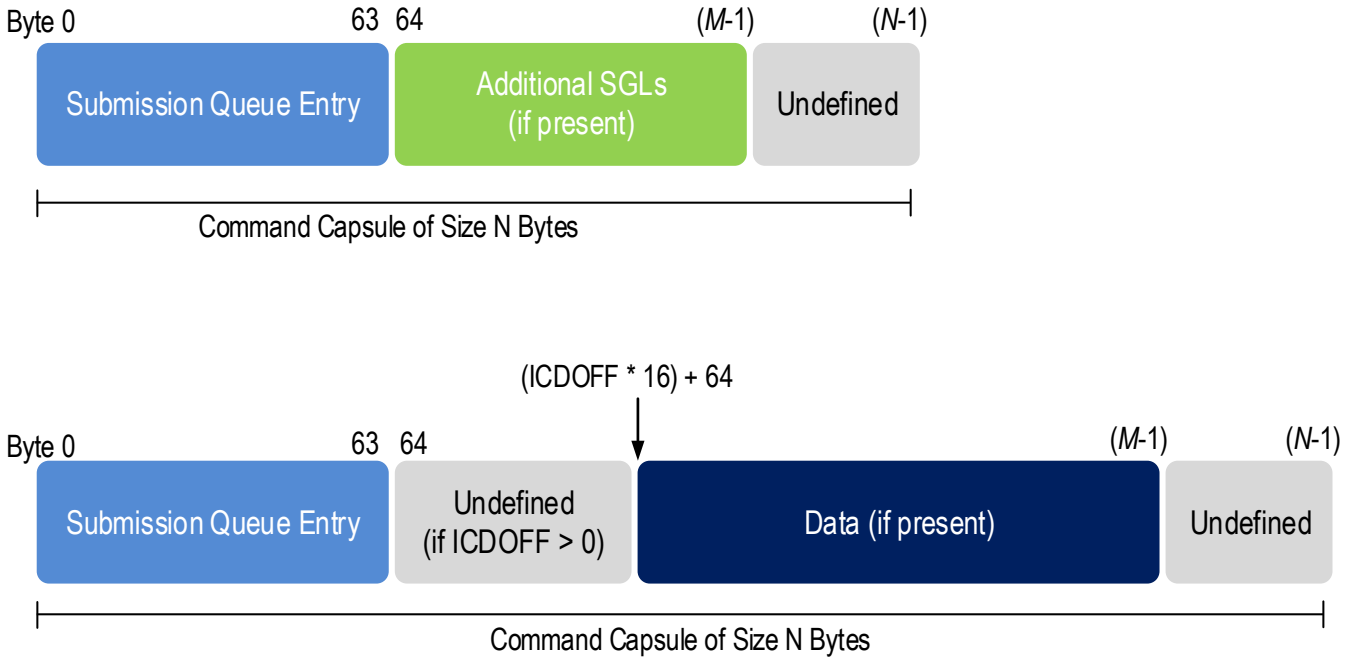
An NVMe Transport binding specification defines the support for data as part of the command capsule. The controller indicates the starting location of data within a command capsule via the In Capsule Data Offset (ICDOFF) field in the Identify Controller data structure.

There are restrictions for SGLs that the host should follow:

- if the ICDOFF field is a non-zero value, then all SGL descriptors following the submission queue entry shall not have a total size greater than (ICDOFF * 16);
- if the SGL descriptors following the submission queue entry have a total size greater than (ICDOFF * 16), then the controller shall abort the command with a status code of Invalid Number of SGL Descriptors;
- the host shall not place more SGL Data Block or Keyed SGL Data Block descriptors within a capsule than the maximum indicated in the Identify Controller data structure; and
- if the host places more SGL Data Block of Keyed SGL Data Block descriptors in a capsule than the maximum indicated in the Maximum SGL Data Block Descriptors field in the Identify Controller data structure, then the controller shall abort the command with a status code of Invalid Number of SGL Descriptors.

The host shall start data (if present) in command capsules at byte offset (ICDOFF * 16) from the end of the submission queue entry.

Figure 83: Data and SGL Locations within a Command Capsule

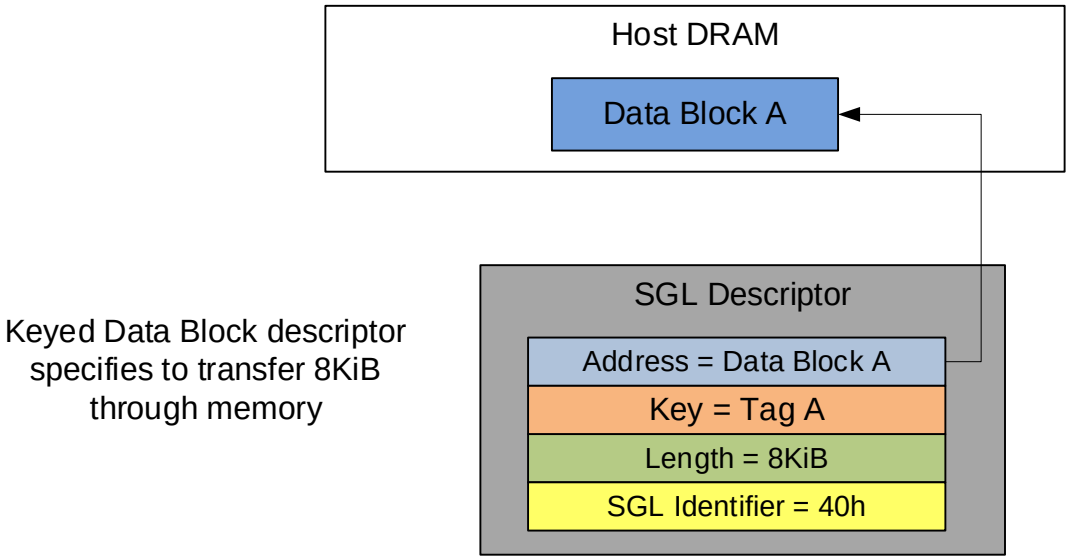


3.3.2.1.3.2 Data Transfer Examples

The data transfer examples in Figure 84 and Figure 85 show SGL examples for a Write command where data is transferred via a memory transaction or within the capsule. The SGL may use a key as part of the data transfer depending on the requirements of the NVMe Transport used.

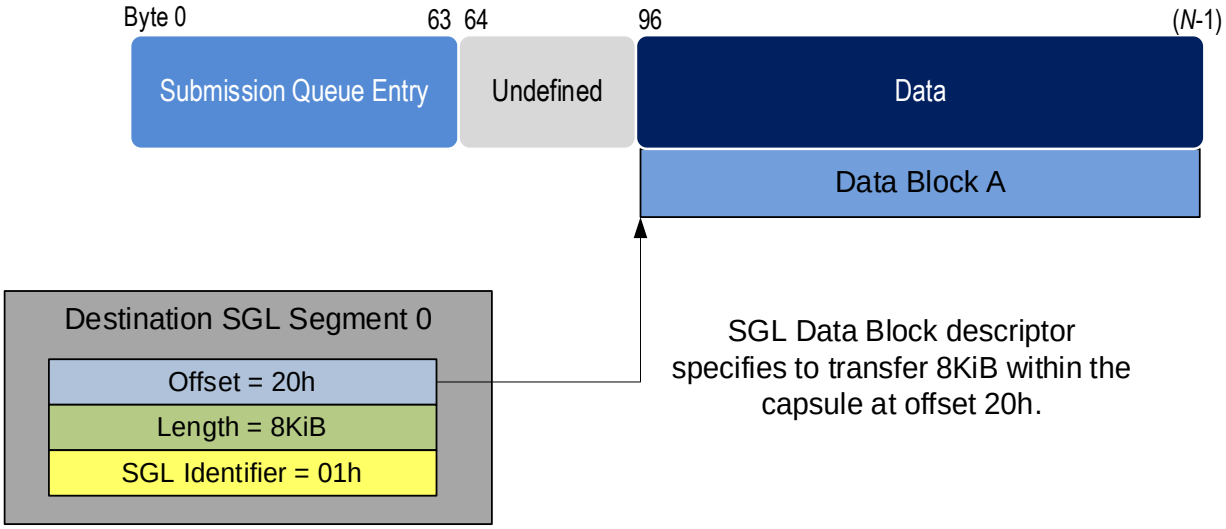
The first example shows an 8KiB write where all of the data is transferred via memory transactions. In this case, there is one SGL descriptor that is contained within the submission queue entry at CMD.SGL1. The SGL descriptor is a Keyed SGL Data Block descriptor. If more SGLs are required to complete the command, the additional SGLs are contained in the command capsule.

Figure 84: SGL Example Using Memory Transactions



The second example shows an 8KiB write where all of the data is transferred within the capsule. In this case, the SGL descriptor is an SGL Data Block descriptor specifying an Offset of 20h based on an ICDOFF value of 2h.

Figure 85: SGL Example Using In Capsule Data Transfer



3.3.2.2 Queue Creation

Controllers using the message-based transport queue model use the Connect command (refer to section 6.3) to create controller Admin or I/O Queues. The creation of an Admin Queue establishes an association between a host and the corresponding controller. The message-based transport queue model does not support the Admin Submission Queue Base Address (ASQ), Admin Completion Queue Base Address (ACQ), and Admin Queue Attributes (AQA) properties as all information necessary to establish an Admin Queue is contained in the Connect command. The message-based transport queue model does not support the Admin commands associated with I/O Queue creation and deletion (Create I/O Completion Queue, Create I/O Submission Queue, Delete I/O Completion Queue, Delete I/O Submission Queue).

An NVMe Transport connection is established between a host and an NVM subsystem prior to the transfer of any capsules or data. The mechanism used to establish an NVMe Transport connection is NVMe Transport specific and defined by the corresponding NVMe Transport binding specification. The NVMe Transport may require a separate NVMe Transport connection for each Admin or I/O Queue or may utilize the same NVMe Transport connection for all Admin and I/O Queues associated with a particular controller. An NVMe Transport may also require that NVMe layer information be passed between the host and controller in the process of establishing an NVMe Transport connection (e.g., exchange queue size to appropriately size send and receive buffers).

The Connect command specifies the Queue ID and type (Admin or I/O), the size of the Submission and Completion Queues, queue attributes, Host NQN, NVM Subsystem NQN, and Host Identifier. The Connect command may specify a particular controller if the NVM subsystem supports a static controller model. The Connect response indicates whether the connection was successfully established as well as whether NVMe in-band authentication is required.

The Connect command is submitted to the same Admin Queue or I/O Queue that the Connect command creates. The underlying NVMe Transport connection that is used for that queue is created first and the Connect command and response capsules are sent over that NVMe Transport connection. The Connect command shall be sent once to a queue.

When a Connect command successfully completes, the corresponding Submission and Completion Queues are created. If NVMe in-band authentication is required as indicated in the Connect response, then

NVMe in-band authentication shall be performed before the queues may be used to perform other Fabrics commands, Admin commands, or I/O commands. Once a Connect command for an Admin Queue has completed successfully (and NVMe in-band authentication, if required, has succeeded), only Fabrics commands may be submitted until the controller is ready (CSTS.RDY = 1). Both Fabrics commands and Admin commands may be submitted to the Admin Queue while the controller is ready. A Connect command for an I/O Queue may be submitted after the controller is ready. Once a Connect command for an I/O Queue has completed successfully (and NVMe in-band authentication, if required, has succeeded), I/O commands may be submitted to the queue.

The Connect response contains the controller ID allocated to the host. All subsequent Connect commands that create an I/O Queue with that controller shall be from the same host, utilize the same NVMe Transport, and have the same Host Identifier, Host NQN, and NVM Subsystem NQN; if any of these conditions are not met, then the Connect command fails.

3.3.2.3 Queue Initialization and Queue State

When a Connect command successfully completes, the corresponding Admin Submission and Completion Queue or I/O Submission and Completion Queues are created. If the host sends a Connect command specifying the Queue ID of a queue which already exists, then the controller shall abort the command with a status code of Command Sequence Error.

The Authentication Requirements (AUTHREQ) field in the Connect response indicates if NVMe in-band authentication is required. If AUTHREQ is cleared to 0h, the created queue is ready for use after the Connect command completes successfully. If AUTHREQ is set to a non-zero value, the created queue is ready for use after NVMe in-band authentication has been performed successfully using the Authentication Send and Authentication Receive Fabrics commands.

If a controller requires or is undergoing NVMe in-band authentication for a queue pair, then a controller shall abort all commands received on that queue other than authentication commands with a status code of Authentication Required. After the NVMe in-band authentication has been performed successfully on a queue, then a controller shall abort all authentication commands on that queue with a status code of Command Sequence Error.

When an Admin Queue is first created, the associated controller is disabled (i.e., CC.EN is initialized to ‘0’). A disabled controller shall abort all commands other than Fabrics commands on the Admin Queue with a status code of Command Sequence Error. After the controller is enabled, it shall accept all supported Admin commands in addition to Fabrics commands.

A created I/O queue shall abort all commands with a status code of Command Sequence Error if the associated controller is disabled.

3.3.2.4 I/O Queue Deletion

NVMe over Fabrics deletes an individual I/O Queue and may delete the associated NVMe Transport connection as a result of:

- the exchange of a Disconnect command and response (refer to section 6.4) between a host and controller; or
- the detection and processing of a transport error on an NVMe Transport connection.

The host indicates support for the deletion of an individual I/O Queue by setting bit 3 to ‘1’ in the CATTR field in the Connect command (refer to Figure 380) used to create the Admin Queue. The controller indicates support for the deletion of an individual I/O Queue by setting bit 0 to ‘1’ in the OFCS field in the Identify Controller Attributes region of the Identify Controller data structure (refer to Figure 275).

If both the host and the controller support deletion of an individual I/O Queue, then the termination of an individual I/O Queue impacts only that I/O Queue (i.e., the association and all other I/O Queues and their associated NVMe Transport connections are not impacted). If either the host or the controller does not support deletion of an individual I/O Queue, then the deletion of an individual I/O Queue or the termination of an NVMe Transport connection causes the association to be terminated.

NVMe over Fabrics uses the Disconnect command to delete an Individual I/O Queue. This command is sent on the I/O Submission Queue to be deleted and affects only that I/O Submission Queue and its associated I/O Completion Queue (i.e., other I/O Queues are not affected). To delete an I/O Queue, the NVMe Transport connection for that I/O Queue is used. If all Queues associated with an NVMe Transport connection are deleted, then the NVMe Transport connection may be deleted after completion of the Disconnect command. Actions necessary to delete the NVMe Transport connection are transport specific. The association between the host and the controller is not affected.

If a Disconnect command returns a status code other than success, the host may delete an I/O Queue using other methods including:

- waiting a vendor specific amount of time and retry the Disconnect command;
- deleting the NVMe Transport connection (note: this may impact other I/O Queues);
- performing a Controller Level Reset (note: this impacts other I/O Queues); or
- ending the host to controller association.

If the transport requires a separate NVMe Transport connection for each Admin and I/O Queue (refer to section 3.3.2.2), then the host should not delete an NVMe Transport connection until after:

- a Disconnect command has been submitted to the I/O Submission Queue; and
- the response for that Disconnect command has been received by the host on the corresponding I/O Completion Queue or a vendor specific timeout (refer to section 3.9) has occurred while waiting for that response.

If the transport requires a separate NVMe Transport connection for each Admin and I/O Queue, then the controller should not delete an NVMe Transport connection until after:

- a Disconnect command has been received on the I/O Submission Queue and processed by the controller;
- the responses for commands received by the controller on that I/O Submission Queue prior to receiving the Disconnect command have been sent to the host on the corresponding I/O Completion Queue; and
- the resulting response for that Disconnect command has been sent to the host on the corresponding I/O Completion queue (i.e., this response is the last response sent). It is recommended that the controller delay destroying the NVMe Transport connection to allow time for the Disconnect command response to be received by the host (e.g., a transport specific event occurs or a transport specific time period elapses).

If the transport utilizes the same NVMe Transport connection for all Admin and I/O Queues associated with a particular controller (refer to section 3.3.2.2), then the deletion of an individual I/O Queue has no impact on the NVMe Transport connection.

A Disconnect command is the last I/O Submission Queue entry processed by the controller for an I/O Queue. Controller processing of the Disconnect command completes or aborts all commands on the I/O Queue on which the Disconnect command was received. The controller determines whether to complete or abort each of those commands.

The response to the Disconnect command is the last I/O Completion Queue entry processed by the host for an I/O Queue. To avoid command aborts the host should wait for outstanding commands on an I/O Queue to complete before sending the Disconnect command.

If the controller detects an NVMe Transport connection loss, then the controller shall stop processing all commands received on I/O Queues associated with that NVMe Transport connection. Until the controller detects an NVMe Transport connection loss or sends a successful completion for a Disconnect command, outstanding commands may continue being processed by the controller.

If the host detects an NVMe Transport connection loss before the responses are received for all outstanding commands submitted to the associated I/O Queue, then there is no further information available to the host about the state of those commands (e.g., each individual outstanding command may have been completed or aborted by the controller).

If an NVMe Transport connection is lost as a result of an NVMe Transport error, then before performing recovery actions related to commands sent on I/O queues associated with that NVMe Transport connection, the host should wait for at least the longer of:

- the NVMe Keep Alive timeout; or
- the underlying fabric transport timeout, if any.

3.3.2.5 Submission Queue Flow Control Negotiation

Use of Submission Queue (SQ) flow control is negotiated for each queue pair by the Connect command and the controller response to the Connect command. SQ flow control shall be used unless it is disabled as a result of that negotiation. If SQ flow control is disabled, then the Submission Queue Head Pointer (SQHD) field is reserved in all Fabrics response capsules for that queue pair after the response to the Connect command (i.e., in all subsequent response capsules for that queue pair, the controller shall clear the SQHD field to 0h and the host should ignore the SQHD field).

If the host requests that SQ flow control be disabled for a queue pair, then the host should size each Submission Queue to support the maximum number of commands that the host could have outstanding at one time for that Submission Queue.

The maximum size of the Admin Submission Queue is specified in the Admin Max SQ Size (ASQSZ) field of the Discovery Log Page Entry for the NVM subsystem (refer to section 5.16.1.23).

The maximum size of an I/O Submission Queue is specified in the Maximum Queue Entries Supported (MQES) field of the Controller Capabilities (CAP) property for the controller (refer to section 3.1.3.1).

The Maximum Outstanding Commands (MAXCMD) value in the Identify Controller data structure indicates the maximum number of commands that the controller processes at one time for a particular I/O Queue. The host may use this value to size I/O Submission Queues and optimize the number of commands submitted at one time per queue to achieve the best performance.

If SQ flow control is disabled, then the host should limit the number of outstanding commands for a queue pair to be less than the size of the Submission Queue. If the controller detects that the number of outstanding commands for a queue pair is greater than or equal to the size of the Submission Queue, then the controller shall:

a) stop processing commands and set the Controller Fatal Status (CSTS.CFS) bit to ‘1’ (refer to section 9.5); and
b) terminate the NVMe Transport connection and end the association between the host and the controller.

3.3.2.6 Submission Queue Flow Control

This section applies only to Submission Queues that use SQ flow control.

The Submission Queue has a Head entry pointer and a Tail entry pointer that are used to manage the queue and determine the number of Submission Queue capsules available to the host for new submissions. The Head and Tail entry pointers are initialized to 0h when a queue is created. All arithmetic operations and comparisons on entry pointers are performed modulo the queue size with queue wrap conditions taken into account. The host increments the Tail entry pointer when the host adds a capsule to a queue. The controller increments the Head entry pointer when that controller removes a capsule from the queue.

The Submission Queue Head entry pointer is maintained by the controller and is communicated to the host in the SQHD field of completion queue entries. The host uses the received SQHD values for Submission Queue management (e.g., to determine whether the Submission Queue is full).

The Submission Queue Tail entry pointer is local to the host and is not communicated to the controller.

The Submission Queue is full when the Head entry pointer equals one more than the Tail entry pointer (i.e., incrementing the Tail entry pointer has caused it to wrap around to just behind the Head entry pointer). A full Submission Queue contains one less capsule than the queue size. A host may continue to submit commands to a Submission Queue as long as the queue is not full.

If the controller detects that the host has submitted a command capsule to a full Submission Queue, then the controller shall:

a) stop processing commands and set the Controller Fatal Status (CSTS.CFS) bit to '1' (refer to section 9.5); and
b) terminate the NVMe Transport connection and end the association between the host and the controller.

The Submission Queue is empty when the Head entry pointer equals the Tail entry pointer.

3.3.2.7 Submission Queue Head Pointer Update Optimization

Submission Queue Head Pointer update optimization does not apply to queue pairs for which Submission Queue (SQ) flow control is disabled, as the SQHD field is reserved if SQ flow control is disabled, refer to section 3.3.2.5 and to section 6.3.

The NVMe Transport may omit transmission of the SQHD value for a response capsule that:

a) contains a Generic Command status (i.e., Status Code Type 0h) indicating successful completion of a command (i.e., Status Code 00h);
b) is not a Connect response capsule; and
c) is not a Disconnect response capsule.

If a new SQHD value is not received in a response capsule, the host continues to use its previous SQHD value. Thus, at the NVMe layer there is a logical progression of SQHD values despite the fact that the NVMe Transport may not actually transfer the SQHD value in each response capsule.

The NVMe Transport may deliver response capsules that do not contain an SQHD value to the host in any order. The applicable NVMe Transport binding specification defines how presence versus absence of an SQHD value in a response capsule is indicated by the NVMe Transport.

Periodic SQHD updates at the host are required to avoid Submission Queue (SQ) starvation as SQHD value transmission in responses is the only means of releasing SQ slots for host reuse.

An NVMe Transport may transmit an SQHD value in every response capsule. If an NVMe Transport does not transmit an SQHD value in every response capsule, then an SQHD value should be transmitted periodically (e.g., in at least one of every n response capsules on a CQ, where n is 10% of the size of the associated SQ) or more often. An SQHD value should always be transmitted if 90% or more of the slots in the associated SQ are occupied at the subsystem.

3.3.2.8 Completion Queue Considerations

Completion Queue flow control (refer to section 3.3.1.2.1) is not used in the message-based transport queue model. Message-based transport Completion Queues do not use either Head entry pointers or Tail entry pointers.

The host should size each Completion Queue to support the maximum number of commands that the host could have outstanding at one time for a particular Submission Queue. The Completion Queue size may be larger than the size of the corresponding Submission Queue to accommodate responses for commands that are being processed by the controller in addition to responses for commands that are still in the Submission Queue.

If the size of a Completion Queue is too small for the number of outstanding commands and the controller submits a response capsule to a full Completion Queue, then the results are undefined.

The Maximum Outstanding Commands (MAXCMD) value in the Identify Controller data structure indicates the maximum number of commands that the controller processes at one time for a particular I/O Queue. The host may use this value to size I/O Completion Queues and optimize the number of commands submitted at one time per queue to achieve the best performance.

Altering a response capsule between controller submission to the Completion Queue and transport delivery of that capsule to the host results in undefined behavior.

3.3.2.9 Transport Requirements

This section defines requirements that all NVMe Transports that support an NVMe over Fabrics implementation shall meet.

The NVMe Transport may support NVMe Transport error detection and report errors to the NVMe layer in command status values. The controller may record NVMe Transport specific errors in the Error Information log page. Transport errors that cause loss of a message or loss of data in a way that the low-level NVMe Transport cannot replay or recover should cause:

- the deletion of the individual I/O Queues (refer to section 3.3.2.4) and the associated NVMe Transport connection on which that NVMe Transport level error occurred; or
- termination of the NVMe Transport connection and the association between the host and controller.

The NVMe Transport shall provide reliable delivery of capsules between a host and NVM subsystem (and allocated controller) over each connection. The NVMe Transport may deliver command capsules in any order on each queue except for I/O commands that are part of fused operations (refer to section 3.4.2).

For command capsules that are part of fused operations for I/O commands, the NVMe Transport:

a) shall deliver the first and second command capsules for each fused operation to the queue in-order; and
b) shall not deliver any other command capsule for the same Submission Queue between delivery of the two command capsules for a fused operation.

The NVMe Transport shall provide reliable delivery of response capsules from an NVMe subsystem to a host over each connection. The NVMe Transport shall deliver response capsules that include an SQ Head Pointer (SQHD) value to the host in-order; this includes all Connect response capsules and all Disconnect response capsules.

3.3.3 Queueing Data Structures

3.3.3.1 Submission Queue Entry

Each Common Command Format command is 64 bytes in size.

Command Dword 0, Namespace Identifier, Metadata Pointer, PRP Entry 1, PRP Entry 2, SGL Entry 1, and Metadata SGL Segment Pointer have common definitions for all Admin commands and I/O commands for all I/O Command Sets. Metadata Pointer, PRP Entry 1, PRP Entry 2, and Metadata SGL Segment Pointer are not used by all commands. Command Dword 0 is defined in Figure 86.

Figure 86: Command Dword 0

Bits	Description
31:16	**Command Identifier (CID):** This field specifies a unique identifier for the command when combined with the Submission Queue identifier. The value of FFFFh should not be used as the Error Information log page (refer to section 5.16.1.2) uses this value to indicate an error is not associated with a particular command.

Figure 86: Command Dword 0

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>15:14</td><td>PRP or SGL for Data Transfer (PSDT): This field specifies whether PRPs or SGLs are used for any data transfer associated with the command. PRPs shall be used for all Admin commands for NVMe over PCIe implementations. SGLs shall be used for all Admin and I/O commands for NVMe over Fabrics implementations (i.e., this field set to 01b). An NVMe Transport may support only specific values (refer to the applicable NVMe Transport binding specification for details).

<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>PRPs are used for this transfer.</td></tr>
<tr><td>01b</td><td>SGLs are used for this transfer. If used, Metadata Pointer (MPTR) contains an address of a single contiguous physical buffer.

Refer to bit 17 of the SGLS field in the Identify Controller data structure (refer to Figure 275) for alignment requirements.</td></tr>
<tr><td>10b</td><td>SGLs are used for this transfer. If used, Metadata Pointer (MPTR) contains an address of an SGL segment containing exactly one SGL Descriptor that is qword aligned.</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table>

If there is metadata that is not interleaved with the user data, as specified in the Format NVM command, then the Metadata Pointer (MPTR) field is used to point to the metadata. The definition of the Metadata Pointer field is dependent on the setting in this field. Refer to Figure 87.</td></tr>
<tr><td>13:10</td><td>Reserved</td></tr>
<tr><td>09:08</td><td>Fused Operation (FUSE): In a fused operation, a complex command is created by “fusing” together two simpler commands. Refer to section 3.4.2. This field specifies whether this command is part of a fused operation and if so, which command it is in the sequence.

<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>Normal operation</td></tr>
<tr><td>01b</td><td>Fused operation, first command</td></tr>
<tr><td>10b</td><td>Fused operation, second command</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>07:00</td><td>Opcode (OPC): This field specifies the opcode of the command to be executed.</td></tr>
</table>

The Common Command Format is defined in Figure 87. Any additional I/O Command Set defined in the future may use an alternate command size or format.

SGLs shall not be used for Admin commands in NVMe over PCIe implementations.

Figure 87: Common Command Format

Bytes	Description
03:00	**Command Dword 0 (CDW0):** This field is common to all commands and is defined in Figure 86.
07:04	**Namespace Identifier (NSID):** This field specifies the namespace that this command applies to. If the namespace identifier is not used for the command, then this field shall be cleared to 0h. The value FFFFFFFFh in this field is a broadcast value (refer to section 3.2.1.2), where the scope (e.g., the NVM subsystem, all attached namespaces, or all namespaces in the NVM subsystem) is dependent on the command. Refer to Figure 139 and Figure 390 for commands that support the use of the value FFFFFFFFh in this field. Specifying an inactive namespace identifier (refer to section 3.2.1.4) in a command that uses the namespace identifier shall cause the controller to abort the command with a status code of Invalid Field in Command, unless otherwise specified. Specifying an invalid namespace identifier (refer to section 3.2.1.2) in a command that uses the namespace identifier shall cause the controller to abort the command with a status code of Invalid Namespace or Format, unless otherwise specified. If the namespace identifier is used for the command (refer to Figure 139), the value FFFFFFFFh is not supported for that command, and the host specifies a value of FFFFFFFFh, then the controller should abort the command with a status code of Invalid Field in Command, unless otherwise specified. If the namespace identifier is not used for the command and the host specifies a value from 1h to FFFFFFFFh, then the controller should abort the command with a status code of Invalid Field in Command, unless otherwise specified.

Figure 87: Common Command Format

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>11:08</td><td>Command Dword 2 (CDW2): This field is command specific Dword2.</td></tr>
<tr><td>15:12</td><td>Command Dword 3 (CDW3): This field is command specific Dword3.</td></tr>
<tr><td>23:16</td><td>Metadata Pointer (MPTR): If CDW0.PSDT (refer to Figure 86) is cleared to 00b, then this field shall contain the address of a contiguous physical buffer of metadata and that address shall be dword aligned (i.e., bits 1:0 cleared to 00b). The controller is not required to check that bits 1:0 are cleared to 00b. The controller may report an error of Invalid Field in Command if bits 1:0 are not cleared to 00b. If the controller does not report an error of Invalid Field in Command, then the controller shall operate as if bits 1:0 are cleared to 00b.

If CDW0.PSDT is set to 01b, then this field shall contain the address of a contiguous physical buffer of metadata. Refer to bit 17 of the SGLS field in the Identify Controller data structure for alignment requirements.

If CDW0.PSDT is set to 10b, then this field shall contain the address of an SGL segment that contains exactly one SGL Descriptor. The address of that SGL segment shall be qword aligned (i.e., bits 2:0 cleared to 000b). The SGL Descriptor contained in that SGL segment is the first SGL Descriptor of the metadata for the command. If the SGL Descriptor contained in that SGL segment is an SGL Data Block descriptor, then that SGL Data Block Descriptor is the only SGL Descriptor and therefore describes the entire metadata data transfer. Refer to section 4.1.2. The controller is not required to check that bits 2:0 are cleared to 000b. The controller may report an error of Invalid Field in Command if bits 2:0 are not cleared to 000b. If the controller does not report an error of Invalid Field in Command, then the controller shall operate as if bits 2:0 are cleared to 000b.</td></tr>
<tr><td>39:24</td><td>Data Pointer (DPTR): This field specifies the data used in the command.
If CDW0.PSDT is cleared to 00b, then the definition of this field is:
<table>
<tr><td>39:32</td><td>PRP Entry 2 (PRP2): This field:

• is reserved if the data transfer does not cross a memory page boundary;
• specifies the Page Base Address of the second memory page if the data transfer crosses exactly one memory page boundary. E.g.,:
o the command data transfer length is equal in size to one memory page and the offset portion of the PBAO field of PRP1 is non-zero; or
o the Offset portion of the PBAO field of PRP1 is equal to 0h and the command data transfer length is greater than one memory page and less than or equal to two memory pages in size;

and

• is a PRP List pointer if the data transfer crosses more than one memory page boundary. E.g.,:
o the command data transfer length is greater than or equal to two memory pages in size but the offset portion of the PBAO field of PRP1 is non-zero; or
o the command data transfer length is equal in size to more than two memory pages and the Offset portion of the PBAO field of PRP1 is equal to 0h.</td></tr>
<tr><td>31:24</td><td>PRP Entry 1 (PRP1): This field contains the first PRP entry for the command or a PRP List pointer depending on the command.</td></tr>
</table>
If CDW0.PSDT is set to 01b or 10b, then the definition of this field is:
<table>
<tr><td>39:24</td><td>SGL Entry 1 (SGL1): This field contains the first SGL segment for the command. If the SGL segment is an SGL Data Block or Keyed SGL Data Block or Transport SGL Data Block descriptor, then it describes the entire data transfer. If more than one SGL segment is needed to describe the data transfer, then the first SGL segment is a Segment, or Last Segment descriptor. Refer to section 4.1.2 for the definition of SGL segments and descriptor types.

The NVMe Transport may support a subset of SGL Descriptor types and features as defined in the NVMe Transport binding specification.</td></tr>
</table></td></tr>
<tr><td>43:40</td><td>Command Dword 10 (CDW10): This field is command specific Dword 10.</td></tr>
<tr><td>47:44</td><td>Command Dword 11 (CDW11): This field is command specific Dword 11.</td></tr>
</table>

Figure 87: Common Command Format

Bytes	Description
51:48	**Command Dword 12 (CDW12):** This field is command specific Dword 12.
55:52	**Command Dword 13 (CDW13):** This field is command specific Dword 13.
59:56	**Command Dword 14 (CDW14):** This field is command specific Dword 14.
63:60	**Command Dword 15 (CDW15):** This field is command specific Dword 15.

In addition to the fields commonly defined for the Common Command Format, Admin and NVM Vendor Specific commands may support the Number of Dwords in Data Transfer and Number of Dwords in Metadata Transfer fields. If supported, the command format for the Admin Vendor Specific Command and NVM Vendor Specific Commands are defined in Figure 88. For more details, refer to section 8.23.

Figure 88: Common Command Format – Admin and NVM Vendor Specific Commands (Optional)

Bytes	Description
03:00	**Command Dword 0 (CDW0):** This field is common to all commands and is defined in Figure 86.
07:04	**Namespace Identifier (NSID):** This field indicates the namespace ID that this command applies to. If the namespace ID is not used for the command, then this field shall be cleared to 0h. Setting this value to FFFFFFFFh causes the command to be applied to all namespaces attached to the controller processing the command, unless otherwise specified. The behavior of a controller in response to an inactive namespace ID for a vendor specific command is vendor specific. Specifying an invalid namespace ID in a command that uses the namespace ID shall cause the controller to abort the command with a status code of Invalid Namespace or Format, unless otherwise specified.
15:08	Reserved
39:16	Refer to Figure 87 for the definition of these fields.
43:40	**Number of Dwords in Data Transfer (NDT):** This field indicates the number of dwords in the data transfer.
47:44	**Number of Dwords in Metadata Transfer (NDM):** This field indicates the number of dwords in the metadata transfer.
51:48	**Command Dword 12 (CDW12):** This field is command specific Dword 12.
55:52	**Command Dword 13 (CDW13):** This field is command specific Dword 13.
59:56	**Command Dword 14 (CDW14):** This field is command specific Dword 14.
63:60	**Command Dword 15 (CDW15):** This field is command specific Dword 15.

3.3.3.2 Common Completion Queue Entry

The Common Completion Queue Entry Layout is at least 16 bytes in size. Figure 89 describes the layout of the first 16 bytes of the completion queue entry data structure which follows the Common Completion Queue Entry Layout. The contents of Dword 0 and Dword 1 are command specific. If a command uses Dword 0 or Dword 1, then the definition of these dwords is contained within the associated command definition. If a command does not use Dword 0 or Dword 1, then the unused field(s) are reserved. Dword 2 is defined in Figure 90 and Dword 3 is defined in Figure 91.

If a completion queue entry is constructed via multiple writes, the Phase Tag bit shall be updated in the last write of that completion queue entry.

Figure 89: Common Completion Queue Entry Layout – Admin and All I/O Command Sets

	31	23	15	7 0
DW0	Command Specific			
DW1	Command Specific			
DW2	SQ Identifier		SQ Head Pointer	
DW3	Status	P	Command Identifier	

Figure 90: Completion Queue Entry: DW 2

Bits	Description
31:16	**SQ Identifier (SQID):** Indicates the Submission Queue to which the associated command was issued. This field is used by host software when more than one Submission Queue shares a single Completion Queue to uniquely determine the command completed in combination with the Command Identifier (CID). This is a reserved field in NVMe over Fabrics implementations.
15:00	**SQ Head Pointer (SQHD):** Indicates the current Submission Queue Head pointer for the Submission Queue indicated in the SQ Identifier field. This is used to indicate to the host the submission queue entries that have been consumed and may be re-used for new entries. Note: The value returned is the value of the SQ Head pointer when the completion queue entry was created. By the time host software consumes the completion queue entry, the controller may have an SQ Head pointer that has advanced beyond the value indicated.

Figure 91: Completion Queue Entry: DW 3

Bits	Description
31:17	**Status :** Indicates the status for the command that is being completed. Refer to section 3.3.3.2.1.
16	**Phase Tag (P):** Identifies whether a completion queue entry is new. Refer to section 3.3.3.2.2. This is a reserved bit in NVMe over Fabrics implementations.
15:00	**Command Identifier (CID):** Indicates the identifier of the command that is being completed. This identifier is assigned by host software when the command is submitted to the Submission Queue. The combination of the SQ Identifier and Command Identifier uniquely identifies the command that is being completed. The maximum number of requests outstanding for a Submission Queue at one time is 64 Ki.

3.3.3.2.1 Status Field Definition

The Status field defines the status for the command indicated in the completion queue entry, defined in Figure 92.

A value of 0h for the Status field indicates a successful command completion, with no fatal or non-fatal error conditions. Unless otherwise noted, if a command fails to complete successfully for multiple reasons, then the particular status code returned is chosen by the vendor.

Figure 92: Completion Queue Entry: Status Field

Bits	Description
31	**Do Not Retry (DNR):** If set to '1', indicates that if the same command is re-submitted to any controller in the NVM subsystem, then that re-submitted command is expected to fail. If cleared to '0', indicates that the same command may succeed if retried. If a command is aborted due to time limited error recovery (refer to the Error Recovery section in the NVM Command Set Specification), this bit should be cleared to '0'. If the SCT and SC fields are cleared to 0h, then this bit should be cleared to '0'.
30	**More (M):** If set to '1', there is more status information for this command as part of the Error Information log page that may be retrieved with the Get Log Page command. If cleared to '0', there is no additional status information for this command. Refer to section 5.16.1.2.

Figure 92: Completion Queue Entry: Status Field

Bits	Description
29:28	**Command Retry Delay (CRD):** If the DNR bit is cleared to '0' and the host has set the Advanced Command Retry Enable (ACRE) field to 1h in the Host Behavior Support feature (refer to section 5.27.1.18), then: a) a 00b CRD value indicates a command retry delay time of zero (i.e., the host may retry the command immediately); and b) a non-zero CRD value selects a field in the Identify Controller data structure (refer to Figure 275) that indicates the command retry delay time: • a 01b CRD value selects the Command Retry Delay Time 1 (CRDT1) field; • a 10b CRD value selects the Command Retry Delay Time 2 (CRDT2) field; and • a 11b CRD value selects the Command Retry Delay Time 3 (CRDT3) field. The host should not retry the command until at least the amount of time indicated by the selected field has elapsed. It is not an error for the host to retry the command prior to that time. If the DNR bit is set to'1' in the Status field or the ACRE field is cleared to 0h in the Host Behavior Support feature, then this field is reserved. If the SCT and SC fields are cleared to 0h, then this field should be cleared to 00b.
27:25	**Status Code Type (SCT):** Indicates the status code type of the completion queue entry. This indicates the type of status code the controller is returning.
24:17	**Status Code (SC):** Indicates a status code identifying any error or status information for the command indicated.

Completion queue entries indicate a Status Code Type (SCT) for the type of completion being reported. Figure 93 specifies the status code type values and descriptions.

Figure 93: Status Code – Status Code Type Values

Value	Description	Reference
0h	**Generic Command Status:** Indicates that the command specified by the Command and Submission Queue identifiers in the completion queue entry has completed. These status values are generic across all command types, and include such conditions as success, opcode not supported, and invalid field.	3.3.3.2.1.1
1h	**Command Specific Status:** Indicates a status value that is specific to a particular command opcode. These values may indicate additional processing is required. Status values such as invalid firmware image or exceeded maximum number of queues is reported with this type.	3.3.3.2.1.2
2h	**Media and Data Integrity Errors:** Any media specific errors that occur in the NVM or data integrity type errors shall be of this type.	3.3.3.2.1.3
3h	**Path Related Status:** Indicates that the command specified by the Command and Submission Queue identifier in the completion queue entry has completed. These status values are generic across all command types. These values may indicate that additional process is required and indicate a status value that is specific to: the connection between the host and the controller processing the command; or the characteristics that support Asymmetric Namespace Access Reporting (refer to section 8.1), the characteristics of the relationship between the controller processing the command and the specified namespace.	3.3.3.2.1.4
4h to 6h	Reserved	
7h	Vendor Specific	

The Status Code (SC) field in the completion queue entry indicates more detailed status information about the completion being reported.

Each Status Code set of values is split into three ranges:

- 00h to 7Fh: Applicable to Admin Command Set, or across multiple command sets;
- 80h to BFh: I/O Command Set specific status codes; and

- C0h to FFh: Vendor Specific status codes.

Unless otherwise specified, if multiple status codes apply, then the controller selects the status code that is returned.

3.3.3.2.1.1 Generic Command Status Definition

Completion queue entries with a Status Code Type (SCT) of Generic Command Status indicate a status value associated with the command that is generic across many different types of commands.

Figure 94: Status Code – Generic Command Status Values

Value	Description	I/O Command Set Specific	I/O Command Set(s)[1]
00h	**Successful Completion:** The command completed without error.	No	
01h	**Invalid Command Opcode:** A reserved coded value or an unsupported value in the command opcode field.	No	
02h	**Invalid Field in Command:** A reserved coded value or an unsupported value in a defined field (other than the opcode field). This status code should be used unless another status code is explicitly specified for a particular condition. The field may be in the command parameters as part of the submission queue entry or in data structures pointed to by the command parameters.	No	
03h	**Command ID Conflict:** The command identifier is already in use. Note: It is implementation specific how many commands are searched for a conflict.	No	
04h	**Data Transfer Error:** Transferring the data or metadata associated with a command had an error.	No	
05h	**Commands Aborted due to Power Loss Notification:** Indicates that the command was aborted due to a power loss notification.	No	
06h	**Internal Error:** The command was not completed successfully due to an internal error. Details on the internal device error should be reported as an asynchronous event. Refer to Figure 145 for Internal Error Asynchronous Event Information.	No	
07h	**Command Abort Requested:** The command was aborted due to an Abort command being received that specified the Submission Queue Identifier and Command Identifier of this command (refer to section 5.1).	No	
08h	**Command Aborted due to SQ Deletion:** The command was aborted due to a Delete I/O Submission Queue request received for the Submission Queue to which the command was submitted.	No	
09h	**Command Aborted due to Failed Fused Command:** The command was aborted due to the other command in a fused operation failing.	No	
0Ah	**Command Aborted due to Missing Fused Command:** The fused command was aborted due to the adjacent submission queue entry not containing a fused command that is the other command in a supported fused operation (refer to section 3.4.2).	No	
0Bh	**Invalid Namespace or Format:** The namespace or the format of that namespace is invalid.	No	
0Ch	**Command Sequence Error:** The command was aborted due to a protocol violation in a multi-command sequence (e.g., a violation of the Security Send and Security Receive sequencing rules in the TCG Storage Synchronous Interface Communications protocol (refer to TCG Storage Architecture Core Specification)).	No	

Figure 94: Status Code – Generic Command Status Values

Value	Description	I/O Command Set Specific	I/O Command Set(s)[1]
0Dh	**Invalid SGL Segment Descriptor:** The command includes an invalid SGL Last Segment or SGL Segment descriptor. This may occur under various conditions, including: **a)** the SGL segment pointed to by an SGL Last Segment descriptor contains an SGL Segment descriptor or an SGL Last Segment descriptor; **b)** an SGL Last Segment descriptor contains an invalid length (i.e., a length of 0h or 1h that is not a multiple of 16); or **c)** an SGL Segment descriptor or an SGL Last Segment descriptor contains an invalid address (e.g., an address that is not qword aligned).	No	
0Eh	**Invalid Number of SGL Descriptors:** There is an SGL Last Segment descriptor or an SGL Segment descriptor in a location other than the last descriptor of a segment based on the length indicated. This is also used for invalid SGLs in a command capsule.	No	
0Fh	**Data SGL Length Invalid:** This may occur if the length of a data SGL is too short. This may occur if the length of a data SGL is too long and the controller does not support SGL transfers longer than the amount of data to be transferred as indicated in the SGL Support field of the Identify Controller data structure.	No	
10h	**Metadata SGL Length Invalid:** This may occur if the length of a metadata SGL is too short. This may occur if the length of a metadata SGL is too long and the controller does not support SGL transfers longer than the amount of data to be transferred as indicated in the SGL Support field of the Identify Controller data structure.	No	
11h	**SGL Descriptor Type Invalid:** The type of an SGL Descriptor is a type that is not supported by the controller, or the combination of type and subtype is not supported by the controller.	No	
12h	**Invalid Use of Controller Memory Buffer:** The attempted use of the Controller Memory Buffer is not supported by the controller. Refer to section 8.3.	No	
13h	**PRP Offset Invalid:** The Offset field for a PRP entry is invalid. This may occur when there is a PRP entry with a non-zero offset after the first entry or when the Offset field in any PRP entry is not dword aligned (i.e., bits 1:0 are not cleared to 00b).	No	
14h	**Atomic Write Unit Exceeded:** See the applicable I/O Command Set specification for the description.	**Yes**	NVM, ZNS
15h	**Operation Denied:** The command was denied due to lack of access rights. Refer to the appropriate security specification (e.g., TCG Storage Interface Interactions specification). For media access commands, the Access Denied status code should be used instead.	No	
16h	**SGL Offset Invalid:** The offset specified in an SGL descriptor is invalid. This may occur when using capsules for data transfers in NVMe over Fabrics implementations and an invalid offset in the capsule is specified.	No	
17h	Reserved	No	
18h	**Host Identifier Inconsistent Format:** The NVM subsystem detected the simultaneous use of 64-bit and 128-bit Host Identifier values on different controllers.	No	
19h	**Keep Alive Timer Expired:** The Keep Alive Timer expired.	No	

Figure 94: Status Code – Generic Command Status Values

Value	Description	I/O Command Set Specific	I/O Command Set(s)[1]
1Ah	**Keep Alive Timeout Invalid:** The Keep Alive Timeout value specified is invalid. This may be due to an attempt to specify a value of 0h on a transport that requires the Keep Alive feature to be enabled. This may be due to the value specified being too large for the associated NVMe Transport as defined in the NVMe Transport binding specification.	No	
1Bh	**Command Aborted due to Preempt and Abort:** The command was aborted due to a Reservation Acquire command with the Reservation Acquire Action (RACQA) set to 010b (Preempt and Abort).	No	
1Ch	**Sanitize Failed:** The most recent sanitize operation failed and no recovery action has been successfully completed.	No	
1Dh	**Sanitize In Progress:** The requested function (e.g., command) is prohibited while a sanitize operation is in progress. Refer to section 8.21.1.	No	
1Eh	**SGL Data Block Granularity Invalid:** See the applicable I/O Command Set specification for the description.	Yes	NVM, ZNS
1Fh	**Command Not Supported for Queue in CMB:** The implementation does not support submission of the command to a Submission Queue in the Controller Memory Buffer or command completion to a Completion Queue in the Controller Memory Buffer. Note: NVM Express revision 1.3 and later use this status code only for Sanitize commands.	No	
20h	**Namespace is Write Protected:** The command is prohibited while the namespace is write protected as a result of a change in the namespace write protection state as defined by the Namespace Write Protection State Machine (refer to Figure 430).	No	
21h	**Command Interrupted:** Command processing was interrupted and the controller is unable to successfully complete the command. The host should retry the command. If this status code is returned, then the controller shall clear the Do Not Retry bit to '0' in the Status field of the CQE (refer to Figure 92). The controller shall not return this status code unless the host has set the Advanced Command Retry Enable (ACRE) field to 1h in the Host Behavior Support feature (refer to section 5.27.1.18).	No	
22h	**Transient Transport Error:** A transient transport error was detected. If the command is retried on the same controller, the command is likely to succeed. A command that fails with a transient transport error four or more times should be treated as a persistent transport error that is not likely to succeed if retried on the same controller.	No	
23h	**Command Prohibited by Command and Feature Lockdown:** The command was aborted due to command execution being prohibited by the Command and Feature Lockdown (refer to section 8.4).	No	

Figure 94: Status Code – Generic Command Status Values

Value	Description	I/O Command Set Specific	I/O Command Set(s)[1]
24h	**Admin Command Media Not Ready:** The Admin command requires access to media and the media is not ready. The Do Not Retry bit indicates whether re-issuing the command at a later time may succeed. This status code shall only be returned: **a)** for Admin commands; and **b)** if the controller is in Controller Ready Independent of Media mode (i.e., CC.CRIME bit is set to '1'). This status code shall not be returned with the Do Not Retry bit cleared to '0' after the amount of time indicated by the Controller Ready With Media Timeout (CRTO.CRWMT) field after the controller is enabled (i.e., CC.EN transitions from '0' to '1'). Refer to Figure 104 for the list of Admin commands permitted to return this status code.	No	
25h to 7Fh	Reserved		
80h	**LBA Out of Range:** See the applicable I/O Command Set specification for the description.	Yes	NVM, ZNS
81h	**Capacity Exceeded:** Execution of the command has caused the capacity of the namespace to be exceeded. This error occurs when the Namespace Utilization exceeds the Namespace Capacity, as reported in Figure 245.	No	
82h	**Namespace Not Ready:** The namespace is not ready to be accessed as a result of a condition other than a condition that is reported as an Asymmetric Namespace Access condition. The Do Not Retry bit indicates whether re-issuing the command at a later time may succeed.	No	
83h	**Reservation Conflict:** The command was aborted due to a conflict with a reservation held on the accessed namespace. Refer to section 8.19.	No	
84h	**Format In Progress:** A Format NVM command is in progress on the namespace. The Do Not Retry bit shall be cleared to '0' to indicate that the command may succeed if resubmitted.	Yes	NVM, ZNS
85h	**Invalid Value Size:** See the applicable I/O Command Set specification for the description.	Yes	KV
86h	**Invalid Key Size:** See the applicable I/O Command Set specification for the description.	Yes	KV
87h	**KV Key Does Not Exist:** See the applicable I/O Command Set specification for the description.	Yes	KV
88h	**Unrecovered Error:** See the applicable I/O Command Set specification for the description.	Yes	KV
89h	**Key Exists:** See the applicable I/O Command Set specification for the description.	Yes	KV
90h to BFh	Reserved		
C0h to FFh	Vendor Specific		

Key:
NVM – NVM Command Set
ZNS – Zoned Namespace Command Set
KV – Key Value Command Set

Notes:
1. This column is blank unless the value is I/O Command Set specific

3.3.3.2.1.2 Command Specific Status Definition

Completion queue entries with a Status Code Type (SCT) of Command Specific Errors indicate an error that is specific to a particular command opcode. Status codes of 00h to 7Fh are for Admin command errors. Status codes of 80h to BFh are specific to the selected I/O command sets.

Figure 95: Status Code – Command Specific Status Values

Value	Description	Commands Affected
00h	Completion Queue Invalid	Create I/O Submission Queue
01h	Invalid Queue Identifier	Create I/O Submission Queue, Create I/O Completion Queue, Delete I/O Completion Queue, Delete I/O Submission Queue
02h	Invalid Queue Size	Create I/O Submission Queue, Create I/O Completion Queue
03h	Abort Command Limit Exceeded	Abort
04h	Reserved	
05h	Asynchronous Event Request Limit Exceeded	Asynchronous Event Request
06h	Invalid Firmware Slot	Firmware Commit
07h	Invalid Firmware Image	Firmware Commit
08h	Invalid Interrupt Vector	Create I/O Completion Queue
09h	Invalid Log Page	Get Log Page
0Ah	Invalid Format	Format NVM, Namespace Management
0Bh	Firmware Activation Requires Conventional Reset	Firmware Commit, Sanitize
0Ch	Invalid Queue Deletion	Delete I/O Completion Queue
0Dh	Feature Identifier Not Saveable	Set Features
0Eh	Feature Not Changeable	Set Features
0Fh	Feature Not Namespace Specific	Set Features
10h	Firmware Activation Requires NVM Subsystem Reset	Firmware Commit, Sanitize
11h	Firmware Activation Requires Controller Level Reset	Firmware Commit, Sanitize
12h	Firmware Activation Requires Maximum Time Violation	Firmware Commit
13h	Firmware Activation Prohibited	Firmware Commit
14h	Overlapping Range	Firmware Commit, Firmware Image Download, Set Features
15h	Namespace Insufficient Capacity	Namespace Management
16h	Namespace Identifier Unavailable	Namespace Management
17h	Reserved	
18h	Namespace Already Attached	Namespace Attachment
19h	Namespace Is Private	Namespace Attachment
1Ah	Namespace Not Attached	Namespace Attachment
1Bh	Thin Provisioning Not Supported	Namespace Management
1Ch	Controller List Invalid	Namespace Attachment
1Dh	Device Self-test In Progress	Device Self-test
1Eh	Boot Partition Write Prohibited	Firmware Commit
1Fh	Invalid Controller Identifier	Virtualization Management
20h	Invalid Secondary Controller State	Virtualization Management
21h	Invalid Number of Controller Resources	Virtualization Management
22h	Invalid Resource Identifier	Virtualization Management
23h	Sanitize Prohibited While Persistent Memory Region is Enabled	Sanitize
24h	ANA Group Identifier Invalid	Namespace Management
25h	ANA Attach Failed	Namespace Attachment
26h	Insufficient Capacity	Capacity Management
27h	Namespace Attachment Limit Exceeded	Namespace Attachment
28h	Prohibition of Command Execution Not Supported	Lockdown
29h	I/O Command Set Not Supported	Namespace Attachment, Namespace Management
2Ah	I/O Command Set Not Enabled	Namespace Attachment

Figure 95: Status Code – Command Specific Status Values

Value	Description	Commands Affected
2Bh	I/O Command Set Combination Rejected	Set Features
2Ch	Invalid I/O Command Set	Identify, Namespace Management
2Dh	Identifier Unavailable	Capacity Management
2Eh to 6Fh	Reserved	
70h to 7Fh	Directive Specific	NOTE 1
80h to BFh	I/O Command Set Specific	Refer to Figure 96
C0h to FFh	Vendor Specific	
Notes: 1. The Directives Specific range defines Directives specific status values. Refer to section 8.7.		

Figure 96: Status Code – Command Specific Status Values, I/O Commands

Value	Description
80h	Conflicting Attributes
81h	Invalid Protection Information
82h	Attempted Write to Read Only Range
83h	Command Size Limit Exceeded
84h to B7h	Reserved
B8h	Zoned Boundary Error
B9h	Zone Is Full
BAh	Zone Is Read Only
BBh	Zone Is Offline
BCh	Zone Invalid Write
BDh	Too Many Active Zones
BEh	Too Many Open Zones
BFh	Invalid Zone State Transition
Notes: 1. A = All I/O Command Sets, C = Command Set Specific.	

Figure 97: Status Code – Command Specific Status Values, Fabrics Commands

Value	Description	Commands Affected
80h	**Incompatible Format:** The NVM subsystem does not support the record format specified by the host.	Connect, Disconnect
81h	**Controller Busy:** The controller is already associated with a host (Connect command). This value is also returned if there is no available controller (Connect command). The controller is not able to disconnect the I/O Queue at the current time (Disconnect command).	Connect, Disconnect
82h	**Connect Invalid Parameters:** One or more of the command parameters (e.g., Host NQN, Subsystem NQN, Host Identifier, Controller ID, Queue ID) specified are not valid.	Connect
83h	**Connect Restart Discovery:** The NVM subsystem requested is not available. The host should restart the discovery process.	Connect
84h	**Connect Invalid Host:** The host is not allowed to establish an association to any controller in the NVM subsystem or the host is not allowed to establish an association to the specified controller.	Connect
85h	**Invalid Queue Type:** The command was sent on the wrong queue type (e.g., a Disconnect command was sent on the Admin queue).	Disconnect
86h to 8Fh	Reserved	
90h	**Discover Restart:** The snapshot of the records is now invalid or out of date. The host should re-read the Discovery Log Page.	Get Log Page

Figure 97: Status Code – Command Specific Status Values, Fabrics Commands

Value	Description	Commands Affected
91h	**Authentication Required:** NVMe in-band authentication is required and the queue has not yet been authenticated.	NOTE 1
92h to AFh	Reserved	
B0h to BFh	**Transport Specific:** The status values in this range are NVMe Transport specific. Refer to the appropriate NVMe Transport binding specification for the definition of these status values.	
Notes: 1. All commands other than Connect, Authenticate Send, and Authenticate Receive.		

3.3.3.2.1.3 Media and Data Integrity Errors Definition

Completion queue entries with a Status Code Type (SCT) of Media and Data Integrity Errors indicate an error associated with the command that is due to an error associated with the NVM media or a data integrity type error.

Figure 98: Status Code – Media and Data Integrity Error Values

Value	Description	Command Set Specific	Command Set(s)
00h to 7Fh	Reserved		
80h	**Write Fault:** The write data could not be committed to the media.	No	
81h	**Unrecovered Read Error:** The read data could not be recovered from the media.	No	
82h	**End-to-end Guard Check Error:** The command was aborted due to an end-to-end guard check failure.	No	
83h	**End-to-end Application Tag Check Error:** The command was aborted due to an end-to-end application tag check failure.	No	
84h	**End-to-end Reference Tag Check Error:** The command was aborted due to an end-to-end reference tag check failure.	No	
85h	**Compare Failure:** See the NVM Command Set Specification for the description.	Yes	NVM
86h	**Access Denied:** Access to the namespace and/or user data is denied due to lack of access rights. Refer to the appropriate security specification (e.g., TCG Storage Interface Interactions Specification).	No	
87h	**Deallocated or Unwritten Logical Block:** See the NVM Command Set Specification for the description.	Yes	NVM
88h	**End-to-End Storage Tag Check Error:** The command was aborted due to an end-to-end storage tag check failure.	No	
89h to BFh	Reserved		
C0h to FFh	Vendor Specific		
Key: NVM – NVM Command Set			

3.3.3.2.1.4 Path Related Status Definition

Completion queue entries with a Status Code Type (SCT) of Path Related Status (refer to Figure 99) indicate a status value associated with the command that is generic across many different types of commands and applies to a specific connection between the host and controller processing the command or between the controller and the namespace. The command for which this status is returned may be retried on a different controller in the same NVM subsystem if more than one controller is available to the host.

In a multipath environment, unless otherwise specified, errors of this type should be retried using a different path, if one is available.

Figure 99: Status Code – Path Related Status Values

Value	Description
00h	**Internal Path Error:** The command was not completed as the result of a controller internal error that is specific to the controller processing the command. Retries for the request function should be based on the setting of the DNR bit (refer to Figure 92).
01h	**Asymmetric Access Persistent Loss:** The requested function (e.g., command) is not able to be performed as a result of the relationship between the controller and the namespace, NVM Set, or Endurance Group being in the ANA Persistent Loss state (refer to section 8.1.3.4). The command should not be re-submitted to the same controller.
02h	**Asymmetric Access Inaccessible:** The requested function (e.g., command) is not able to be performed as a result of the relationship between the controller and the namespace, NVM Set, or Endurance Group being in the ANA Inaccessible state (refer to section 8.1.3.3). The command should not be re-submitted to the same controller.
03h	**Asymmetric Access Transition:** The requested function (e.g., command) is not able to be performed as a result of the relationship between the controller and the namespace, NVM Set, or Endurance Group transitioning between Asymmetric Namespace Access states (refer to section 8.1.3.5). The requested function should be retried after the transition is complete.
04h to 5Fh	Reserved
Controller detected Pathing errors	
60h	**Controller Pathing Error:** A pathing error was detected by the controller.
61h to 6Fh	Reserved
Host detected Pathing errors	
70h	**Host Pathing Error:** A pathing error was detected by the host.
71h	**Command Aborted By Host:** The command was aborted as a result of host action (e.g., the host disconnected the Fabric connection).
72h to 7Fh	Reserved
Other Pathing errors	
80h to BFh	I/O Command Set Specific
C0h to FFh	Vendor Specific

3.3.3.2.2 Phase Tag

The Phase Tag bit indicates whether a completion queue entry is new. The Phase Tag bit for each completion queue entry in:

- the Admin Completion Queue shall be initialized to ‘0’ by the host prior to setting CC.EN (refer to Figure 46) to ‘1’; and
- an I/O Completion Queue shall be initialized to ‘0’ by the host prior to submitting the Create I/O Completion Queue command for that queue.

When the controller posts a new completion queue entry to the Completion Queue, the controller shall invert the Phase Tag bit in that completion queue entry (i.e., the inverting of the Phase Tag bit enables the host to detect the new completion queue entry).

When a completion queue entry is posted to a completion queue slot in:

- the Admin Queue for the first time after CC.EN is set to ‘1’, the Phase Tag bit for that completion queue entry is set to ‘1’; and
- an I/O Completion Queue for the first time after the Create I/O Completion Queue command completed for that queue, the Phase Tag bit for that completion queue entry is set to ‘1’.

This continues for each completion queue entry that is posted until the controller posts a completion queue entry to the highest numbered completion queue slot and wraps to completion queue slot number 0 as described in section 3.3.1.2. When that queue wrap condition occurs, the Phase Tag bit is then cleared to ‘0’ in each completion queue entry that is posted. This continues until another queue wrap condition occurs. Each time a queue wrap condition occurs, the value of the Phase Tag bit is inverted (i.e., changes from ‘1’ to ‘0’ or changes from ‘0’ to ‘1’).

3.3.3.2.2.1 Phase Tag Example

Figure 100 shows an example of how the Phase Tag bit changes over time as the Controller completes commands and the host processes those completions. This example shows a Completion Queue consisting of 6 entries.

Figure 100: Phase Tag bit Transition Example

T[1]	Condition	Completion Queue Entry/Slot number					
		0	1	2	3	4	5
0	Admin Queue: Host initializes Completion Queue and sets CC.EN to '1' I/O Queue: Host initializes Completion Queue and submits Create I/O Completion Queue command	P(0) (E) HEAD-> TAIL->	P(0) (E)	P(0) (E)	P(0) (E)	P(0) (E)	P(0) (E)
1	Controller has completed 1 command and the host has consumed 0 completions	P(1) HEAD->	P(0) (E) TAIL->	P(0) (E)	P(0) (E)	P(0) (E)	P(0) (E)
2	Controller has completed 6 commands and the host has consumed 2 completions	P(1) (E) TAIL->	P(1) (E)	P(1) HEAD- >	P(1)	P(1)	P(1)
3	Controller has completed 7 commands and the host has consumed 2 completions	P(0)	P(1) (E) TAIL->	P(1) HEAD- >	P(1)	P(1)	P(1)
4	Controller has completed 7 commands and host has consumed 4 completions	P(0)	P(1) (E) TAIL->	P(1) (E)	P(1) (E)	P(1) HEAD- >	P(1)
5	Controller has completed 11 commands and host has consumed 8 completions	P(0) (E)	P(0) (E)	P(0) HEAD- >	P(0)	P(0)	P(1) (E) TAIL->
6	Controller has completed 11 commands and host has consumed 11 completions	P(0) (E)	P(0) (E)	P(0) (E)	P(0) (E)	P(0) (E)	P(1) (E) HEAD-> TAIL->

Key:
- P(0) = Phase Tag bit for this completion queue entry is cleared to the value '0'.
- P(1) = Phase Tag bit for this completion queue entry is set to the value '1'.
- (E) = The Entry/Slot is empty.
- HEAD-> = Completion Queue Head Pointer for this completion queue is set to indicate this slot.
- TAIL-> = Completion Queue Tail Pointer for this completion queue is used within the controller to indicate this slot.

Note:
T = Time sequence.

At time 0, the host initializes the Completion queue (i.e., clearing the Phase Tag bit to '0' in each completion queue entry in the completion queue). For the Admin Completion Queue, the host then sets CC.EN to '1' to enable the controller. For an I/O Completion Queue, the host then sends the Create I/O Completion Queue command. The queue, at this time, is in the Empty condition (refer to section 3.3.1.4).

At time 1, the controller has completed a command, but the host has not consumed that completion queue entry. As a result of the command completion, the Phase Tag bit in completion queue entry 0 has been inverted to '1'. Since no completion queue entries have been consumed, the Completion Queue Head pointer still indicates completion queue entry 0. The controller has updated the internal Completion Queue Tail Pointer to indicate that completion queue slot 1 is the next completion queue slot into which the controller posts a completion queue entry.

At time 2, the controller has completed 5 additional commands (i.e., 6 commands have been completed) and the host has consumed 2 of the completion queue entries. As a result of the 5 additional commands having been completed, the Phase Tag bit has been inverted to '1' in completion queue entry 1 through completion queue entry 5. As a result of 2 completion queue entries having been consumed, the host has updated the Completion Queue Head Pointer to indicate that completion queue entry 0 and completion queue entry 1 have been consumed (i.e., completion queue entry 2 is the next completion queue entry for the host to consume). The controller has updated the internal Completion Queue Tail Pointer to indicate that completion queue slot 0 is the next completion queue slot into which the controller posts a completion queue entry.

At time 3, the controller has completed 1 additional command (i.e., 7 commands have been completed) and no additional completion queue entries have been consumed by the host (i.e., 2 completion queue entries have been consumed). As a result of the additional command having been completed, the Phase Tag bit has been inverted to '0' in completion queue entry 0 (i.e., accounting for the queue wrap condition). The controller has updated the internal Completion Queue Tail Pointer to indicate that completion queue slot 1 is the next completion queue slot into which the controller posts a completion queue entry**.** The queue, at this time, is in the Full condition (refer to section 3.3.1.5).

At time 4, the controller has completed no additional commands (i.e., 7 commands have been completed) and the host has consumed 2 additional completion queue entries (i.e., 4 completion queue entries have been consumed). As a result of 2 additional completion queue entries having been consumed, the host has updated the Completion Queue Head Pointer to indicate that completion queue entry 2 and completion queue entry 3 have now been consumed (i.e., completion queue entry 4 is the next completion queue entry for the host to consume). The controller internal Completion Queue Tail Pointer has not changed.

At time 5, the controller has completed 11 commands and the host has consumed 8 of the completion queue entries. As a result of the 4 additional commands having been completed, the Phase Tag bit has been inverted to '0' in completion queue entry 1 through completion queue entry 4. As a result of the 4 additional completion queue entries having been consumed, the host has updated the Completion Queue Head Pointer to indicate that completion queue entry 5 through completion queue entry 1 (i.e., accounting for the queue wrap condition) have now been consumed (i.e., completion queue entry 2 is the next completion queue entry for the host to consume). The controller has updated the internal Completion Queue Tail Pointer to indicate that completion queue slot 5 is the next completion queue slot into which the controller posts a completion queue entry.

At time 6, the controller has completed 11 commands and the host has consumed all 11 of the completion queue entries. As a result of no new command completions, there are no changes to the Phase Tag bit values. As a result of the 3 additional completion queue entries having been consumed, the host has updated the Completion Queue Head Pointer to indicate that completion queue entry 2 through completion queue entry 4 have now been consumed (i.e., completion queue slot 5 is the next completion queue slot from which the host consumes a completion queue entry). The queue, at this time, is in the Empty condition (refer to section 3.3.1.4).

3.3.3.3 Queue Size

The Queue Size is indicated in a 16-bit 0's based field that indicates the number of slots in the queue. The minimum size for a queue is two slots. The maximum size for either an I/O Submission Queue or an I/O Completion Queue is defined as 64 Ki slots, limited by the maximum queue size supported by the controller that is reported in the CAP.MQES field. The maximum size for the Admin Submission and Admin Completion Queue is defined as 4 Ki slots. One slot in each queue is not available for use due to Head and Tail entry pointer definition.

3.3.3.4 Queue Identifier

Each queue is identified through a 16-bit ID value that is assigned to the queue when it is created. Both I/O Submission Queue identifiers and I/O Completion Queue identifiers are a value from 1 to 65,535.

3.3.3.5 Queue Priority

If the weighted round robin with urgent priority class arbitration mechanism is supported, then host software may assign a queue priority service class of Urgent, High, Medium, or Low. If the weighted round robin with urgent priority class arbitration mechanism is not supported, then the priority setting is not used and is ignored by the controller.

3.3.3.6 Queue Coordination

There is one Admin Queue pair associated with multiple I/O queue pairs. The Admin Submission Queue and Completion Queue are used to carry out functions that impact the entire controller. An I/O Submission Queue and Completion Queue may be used to carry out I/O (read/write) operations and may be distributed across CPU cores and threads.

An Admin command may impact one or more I/O queue pairs. The host should ensure that Admin actions are coordinated with threads that are responsible for the I/O queue pairs to avoid unnecessary error conditions. The details of this coordination are outside the scope of this specification.

3.4 Command Architecture Submission and Completion Mechanism

This section describes the command issue and completion mechanism. It also describes how commands are built by host software and command completion processing.

Commands shall only be submitted by the host when the controller is ready as indicated in the Controller Status property (CSTS.RDY) and after appropriate I/O Submission Queue(s) and I/O Completion Queue(s) have been created.

3.4.1 Command Ordering Requirements

Commands which are not part of a fused operation (refer to section 3.4.2) and which comply with atomic operations requirements (refer to section 3.4.3), are processed as independent entities without reference to other commands submitted to the same I/O Submission Queue or to commands submitted to other I/O Submission Queues. For example, the controller is not responsible for checking the LBA of a NVM Command Set Read command or Write command to ensure any type of ordering between commands. If a Read command is submitted for LBA *x* and there is a Write command also submitted for LBA *x*, there is no guarantee of the order of completion for those commands (the Read command may finish first or the Write command may finish first). If there are ordering requirements between these commands, host software or the associated application is required to enforce that ordering above the level of the controller.

3.4.2 Fused Operations

Fused operations enable a more complex command by “fusing” together two simpler commands. This feature is optional; support for this feature is indicated in the FUSES field in the Identify Controller data structure in Figure 275. In a fused operation, the requirements are:

- The commands shall be executed in sequence as an atomic unit. The controller shall behave as if no other operations have been executed between these two commands;
- The operation ends at the point an error is encountered in either command. If the first command in the sequence failed, then the second command in the sequence shall be aborted. If the second command in the sequence failed, then the completion status of the first command is sequence specific and is defined within the Fused Operation section of the applicable I/O Command Set specification;
- The commands shall be inserted next to each other in the same Submission Queue. If the first command is in the last slot in the Submission Queue, then the second command shall be the first slot in the Submission Queue as part of wrapping around. In the memory-based transport queue model, the Submission Queue Tail doorbell pointer update shall indicate both commands as part of one doorbell update. In the message-based transport queue model, the command capsules shall be submitted in-order.
- To abort the fused operation, the host shall submit an Abort command separately for each of the commands; and

- A completion queue entry is posted by the controller for each of the commands.

Whether a command is part of a fused operation is indicated in the Fused Operation (FUSE) field of Command Dword 0 shown in Figure 86. The FUSE field also indicates whether each command is the first command in the fused operation or the second command in the fused operation. If the FUSE field is set to a non-zero value and the controller does not support the requested fused operation, then the controller should abort the command with a status code of Invalid Field in Command.

Refer to each I/O Command Set specification for applicability and additional details, if any.

3.4.3 Atomic Operations

The definition for atomic operations is command set specific. Refer to each I/O Command Set specification for applicability and additional details, if any.

3.4.4 Command Arbitration

After a command has been submitted to the controller (refer to section 1.5.13), the controller transfers submitted commands into the controller for subsequent processing using a vendor specific algorithm.

A command is being processed when the controller and/or namespace state is being accessed or modified by the command (e.g., a Feature setting is being accessed or modified or a logical block is being accessed or modified).

A command is completed when a completion queue entry for the command has been posted to the corresponding Completion Queue. Upon completion, all controller state and/or namespace state modifications made by that command are globally visible to all subsequently submitted commands.

A candidate command is a submitted command which has been transferred into the controller that the controller deems ready for processing. The controller selects command(s) for processing from the pool of submitted commands for each Submission Queue. The commands that comprise a fused operation shall be processed together and in order by the controller. The controller may select candidate commands for processing in any order. The order in which commands are selected for processing does not imply the order in which commands are completed.

Arbitration is the method used to determine the Submission Queue from which the controller starts processing the next candidate command(s). Once a Submission Queue is selected using arbitration, the Arbitration Burst setting determines the maximum number of commands that the controller may start processing from that Submission Queue before arbitration shall again take place. A fused operation may be considered as one or two commands by the controller.

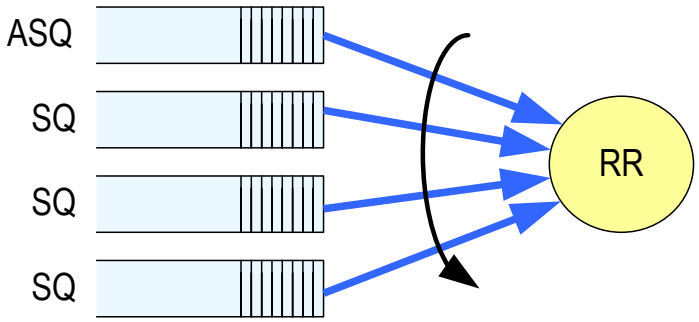
All controllers shall support the round robin command arbitration mechanism. A controller may optionally implement weighted round robin with urgent priority class and/or a vendor specific arbitration mechanism. The Arbitration Mechanism Supported field in the Controller Capabilities property (CC.AMS) indicates optional arbitration mechanisms supported by the controller.

In order to make efficient use of the non-volatile memory, it is often advantageous to execute multiple commands from a Submission Queue in parallel. For Submission Queues that are using weighted round robin with urgent priority class or round robin arbitration, host software may configure an Arbitration Burst setting. The Arbitration Burst setting indicates the maximum number of commands that the controller may launch at one time from a particular Submission Queue. It is recommended that host software configure the Arbitration Burst setting as close to the recommended value by the controller as possible (specified in the Recommended Arbitration Burst field of the Identify Controller data structure in Figure 275), taking into consideration any latency requirements. Refer to section 5.27.1.1.

3.4.4.1 Round Robin Arbitration

If the round robin arbitration mechanism is selected, the controller shall implement round robin command arbitration amongst all Submission Queues, including the Admin Submission Queue. In this case, all Submission Queues are treated with equal priority. The controller may select multiple candidate commands for processing from each Submission Queue per round based on the Arbitration Burst setting.

Figure 101: Round Robin Arbitration



3.4.4.2 Weighted Round Robin with Urgent Priority Class Arbitration

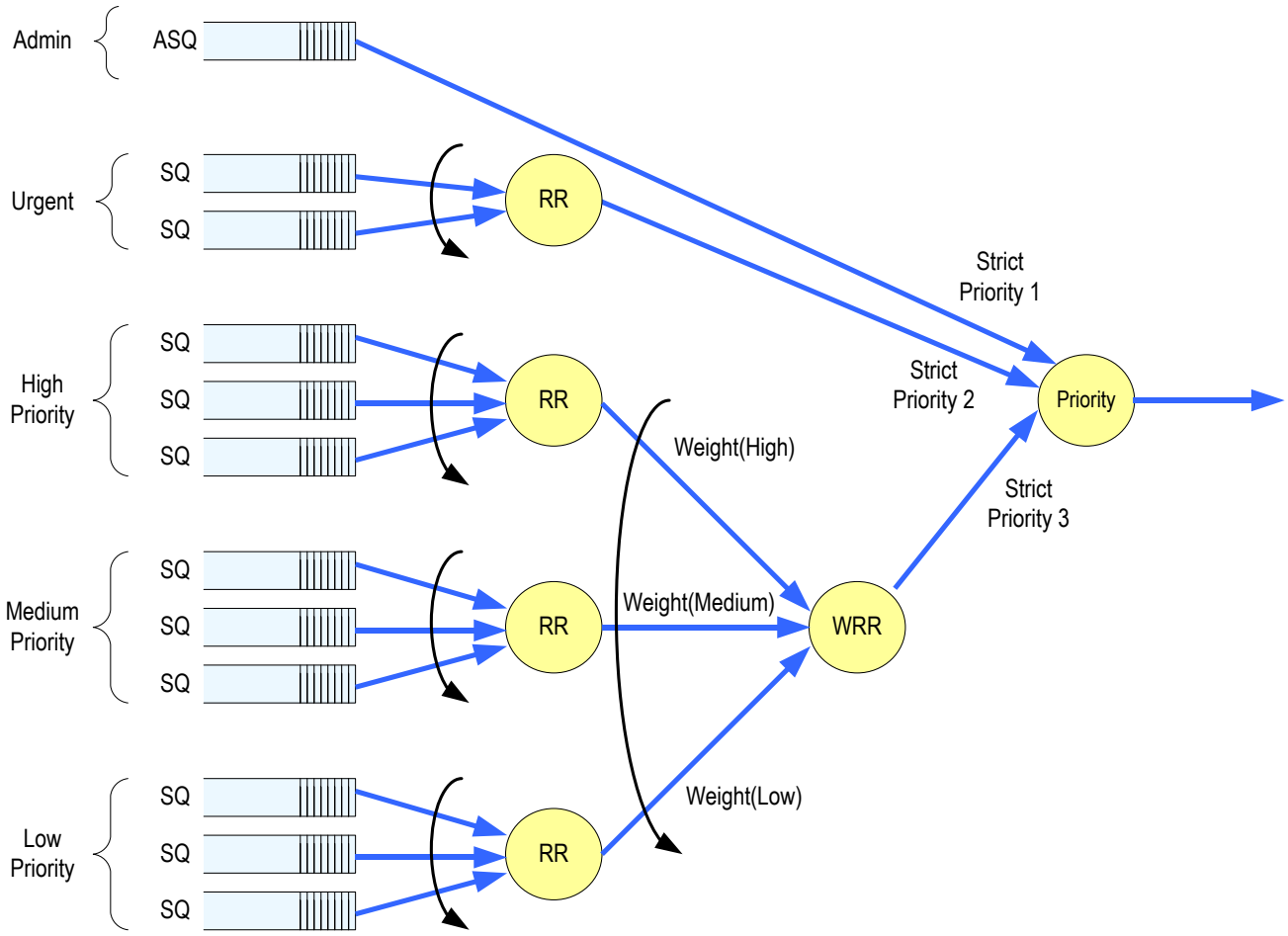
In this arbitration mechanism, there are three strict priority classes and three weighted round robin priority levels. If Submission Queue A is of higher strict priority than Submission Queue B, then all candidate commands in Submission Queue A shall start processing before candidate commands from Submission Queue B start processing.

The highest strict priority class is the Admin class that includes any command submitted to the Admin Submission Queue. This class has the highest strict priority above commands submitted to any other Submission Queue.

The next highest strict priority class is the Urgent class. Any I/O Submission Queue assigned to the Urgent priority class is serviced next after commands submitted to the Admin Submission Queue, and before any commands submitted to a weighted round robin priority level. Host software should use care in assigning any Submission Queue to the Urgent priority class since there is the potential to starve I/O Submission Queues in the weighted round robin priority levels as there is no fairness protocol between Urgent and non Urgent I/O Submission Queues.

The lowest strict priority class is the Weighted Round Robin class. This class consists of the three weighted round robin priority levels (High, Medium, and Low) that share the remaining bandwidth using weighted round robin arbitration. Host software controls the weights for the High, Medium, and Low service classes via the Set Features command. Round robin is used to arbitrate within multiple Submission Queues assigned to the same weighted round robin level. The number of candidate commands that may start processing from each Submission Queue per round is either the Arbitration Burst setting or the remaining weighted round robin credits, whichever is smaller.

Figure 102: Weighted Round Robin with Urgent Priority Class Arbitration



In Figure 102, the Priority decision point selects the highest priority candidate command selected next to start processing.

3.4.4.3 Vendor Specific Arbitration

A vendor may choose to implement a vendor specific arbitration mechanism. The mechanism(s) are outside the scope of this specification.

3.5 Controller Initialization

This section describes the recommended procedure for initializing a controller.

3.5.1 Memory-based Transport Controller Initialization

Upon completion of the transport-specific controller initialization steps defined within the relevant NVMe Transport binding specification, the host should perform the following sequence of actions to initialize the controller to begin executing commands:

1. The host waits for the controller to indicate that any previous reset is complete by waiting for CSTS.RDY to become ‘0’;
2. The host configures the Admin Queue by setting the Admin Queue Attributes (AQA), Admin Submission Queue Base Address (ASQ), and Admin Completion Queue Base Address (ACQ) to appropriate values;

3. The host determines the supported I/O Command Sets by checking the state of CAP.CSS and appropriately initializing CC.CSS as follows:
    a. If the CAP.CSS bit 7 is set to '1', then the CC.CSS field should be set to 111b;
    b. If the CAP.CSS bit 6 is set to '1', then the CC.CSS field should be set to 110b; and
    c. If the CAP.CSS bit 6 is cleared to '0' and bit 0 is set to '1', then the CC.CSS field should be set to 000b;
4. The controller settings should be configured. Specifically:
    a. The arbitration mechanism should be selected in CC.AMS; and
    b. The memory page size should be initialized in CC.MPS;
5. The host enables the controller by setting CC.EN to '1';
6. The host waits for the controller to indicate that the controller is ready to process commands. The controller is ready to process commands when CSTS.RDY is set to '1';
7. The host determines the configuration of the controller by issuing the Identify command specifying the Identify Controller data structure (i.e., CNS 01h);
8. The host determines any I/O Command Set specific configuration information as follows:
    a. If the CAP.CSS bit 6 is set to '1', then the host does the following:
        i. Issue the Identify command specifying the Identify I/O Command Set data structure (CNS 1Ch); and
        ii. Issue the Set Features command with the I/O Command Set Profile Feature Identifier (FID 19h) specifying the index of the I/O Command Set Combination (refer to Figure 289) to be enabled;

        and

    b. For each I/O Command Set that is enabled (Note: the NVM Command Set is enabled if the CC.CSS field is set to 000b):
        i. Issue the Identify command specifying the I/O Command Set specific Active Namespace ID list (CNS 07h) with the appropriate Command Set Identifier (CSI) value of that I/O Command Set; and
        ii. For each NSID that is returned:
            1. If the enabled I/O Command Set is the NVM Command Set or an I/O Command Set based on the NVM Command Set (e.g., the Zoned Namespace Command Set) issue the Identify command specifying the Identify Namespace data structure (CNS 00h); and
            2. Issue the Identify command specifying each of the following data structures (refer to Figure 274): the I/O Command Set specific Identify Namespace data structure, the I/O Command Set specific Identify Controller data structure, and the I/O Command Set independent Identify Namespace data structure;
9. If the controller implements I/O queues, then the host should determine the number of I/O Submission Queues and I/O Completion Queues supported using the Set Features command with the Number of Queues feature identifier. After determining the number of I/O Queues, the NVMe Transport specific interrupt registers (e.g. MSI and/or MSI-X registers) should be configured;
10. If the controller implements I/O queues, then the host should allocate the appropriate number of I/O Completion Queues based on the number required for the system configuration and the number supported by the controller. The I/O Completion Queues are allocated using the Create I/O Completion Queue command;
11. If the controller implements I/O queues, then the host should allocate the appropriate number of I/O Submission Queues based on the number required for the system configuration and the number supported by the controller. The I/O Submission Queues are allocated using the Create I/O Submission Queue command; and
12. To enable asynchronous notification of optional events, the host should issue a Set Features command specifying the events to enable. To enable asynchronous notification of events, the host

should submit an appropriate number of Asynchronous Event Request commands. This step may be done at any point after the controller signals that the controller is ready (i.e., CSTS.RDY is set to '1').

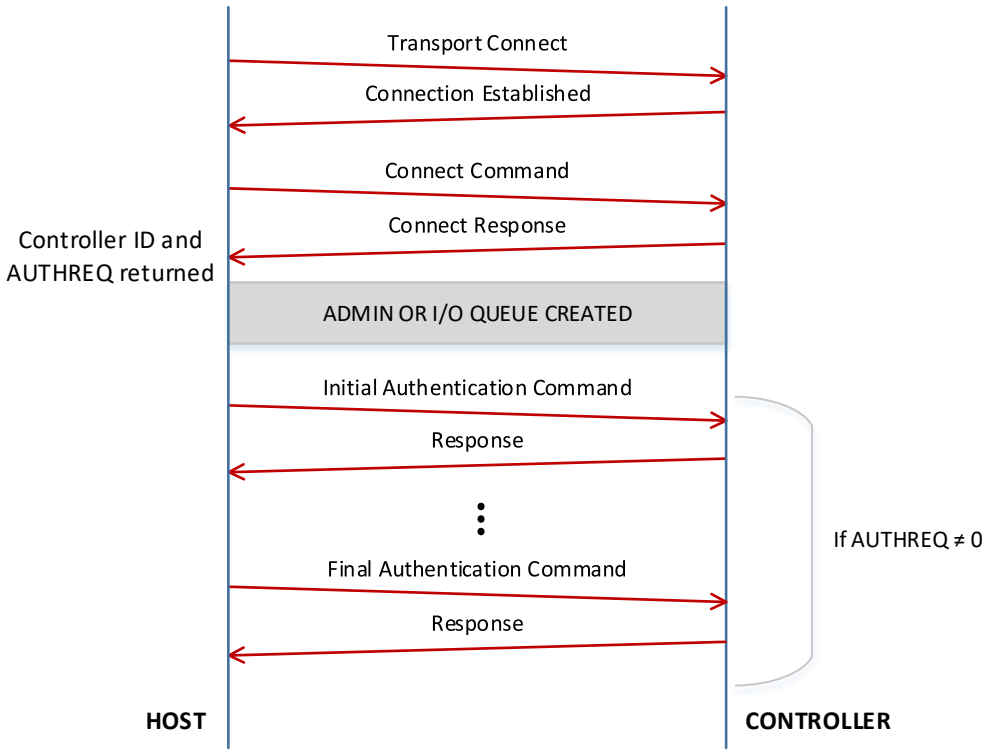
After performing these steps, the controller shall be ready to process Admin or I/O commands issued by the host.

For exit of the D3 power state, the initialization steps outlined should be followed.

3.5.2 Message-based Transport Controller Initialization

The host selects the NVM subsystem with which to create a host to controller association. The host first establishes an NVMe Transport connection with the NVM subsystem. Next the host forms an association with a controller and creates the Admin Queue using the Fabrics Connect command. Finally, the host configures the controller and creates I/O Queues. Figure 103 is a ladder diagram that describes the queue creation process for an Admin Queue or an I/O Queue.

Figure 103: Queue Creation Flow



The controller initialization steps after an association is established are described below. For determining capabilities or configuring properties, the host uses the Property Get and Property Set commands, respectively.

1. NVMe in-band authentication is performed if required (refer to section 8.13.2);
2. The host determines the controller capabilities;
3. The host determines the supported I/O Command Sets by checking the state of CAP.CSS and appropriately initializing CC.CSS as follows:
   a. If the CAP.CSS bit 7 is set to '1', then the CC.CSS field should be set to 111b;
   b. If the CAP.CSS bit 6 is set to '1', then the CC.CSS field should be set to 110b; and
   c. If the CAP.CSS bit 6 is cleared to '0' and bit 0 is set to '1', then the CC.CSS field should be set to 000b;

4. The host configures controller settings. Specific settings include:
    a. The arbitration mechanism should be selected in CC.AMS; and
    b. The memory page size should be initialized in CC.MPS;
5. The controller should be enabled by setting CC.EN to '1';
6. The host should wait for the controller to indicate the controller is ready to process commands. The controller is ready to process commands when CSTS.RDY is set to '1';
7. The host determines the configuration of the controller by issuing the Identify command specifying the Identify Controller data structure (i.e., CNS 01h);
8. The host determines any I/O Command Set specific configuration information as follows:
    a. If the CAP.CSS bit 6 is set to '1', then the host does the following:
        i. Issue the Identify command specifying the Identify I/O Command Set data structure (CNS 1Ch); and
        ii. Issue the Set Features command with the I/O Command Set Profile Feature Identifier (FID 19h) specifying the index of the I/O Command Set Combination (refer to Figure 289) to be enabled;

    and

    b. For each I/O Command Set that is enabled (Note: the NVM Command Set is enabled if the CC.CSS field is set to 000b):
        i. Issue the Identify command specifying the I/O Command Set specific Active Namespace ID list (CNS 07h) with the appropriate Command Set Identifier (CSI) value of that I/O Command Set; and
        ii. For each NSID that is returned:
            1. If the enabled I/O Command Set is the NVM Command Set or an I/O Command Set based on the NVM Command Set (e.g., the Zoned Namespace Command Set) issue the Identify command specifying the Identify Namespace data structure (CNS 00h); and
            2. Issue the Identify command specifying each of the following data structures (refer to Figure 274): the I/O Command Set specific Identify Namespace data structure, the I/O Command Set specific Identify Controller data structure, and the I/O Command Set independent Identify Namespace data structure;
9. The host should determine:
    a. the maximum I/O Queue size using CAP.MQES; and
    b. the number of I/O Submission Queues and I/O Completion Queues supported using the response from the Set Features command with the Number of Queues feature identifier;
10. The host should use the Connect command (refer to section 6.3) to create I/O Submission and Completion Queue pairs; and
11. To enable asynchronous notification of optional events, the host should issue a Set Features command specifying the events to enable. The host may submit one or more Asynchronous Event Request commands to be notified of asynchronous events as described by section 5.2. This step may be done at any point after the controller signals that the controller is ready (i.e., CSTS.RDY is set to '1').

The association may be removed if step 5 (i.e., setting CC.EN to '1') is not completed within 2 minutes after establishing the association.

3.5.3 Controller Ready Modes During Initialization

There are two controller ready modes:

- **Controller Ready With Media:** By the time the controller becomes ready (i.e., by the time that CSTS.RDY transitions from '0' to '1') after the controller is enabled (i.e., CC.EN transitions from '0' to '1'), then:

a) the controller shall be able to process all commands without error as described in section 3.5.4.1; and
b) all namespaces attached to the controller and all media required to process Admin commands shall be ready (i.e., commands are not permitted to be aborted with a status code of Namespace Not Ready with the Do Not Retry bit cleared to '0' or Admin Command Media Not Ready with the Do Not Retry bit cleared to '0').

- **Controller Ready Independent of Media:** After the controller is enabled, all namespaces attached to the controller and media required to process Admin commands may or may not become ready by the time the controller becomes ready. Any NVM command that specifies one or more namespaces attached to the controller is permitted to be aborted with a status code of Namespace Not Ready with the Do Not Retry bit cleared to '0' until CRTO.CRWMT amount of time after the controller is enabled.

  Admin commands that require access to the media are permitted to be aborted with a status code of Admin Command Media Not Ready with the Do Not Retry bit cleared to '0' until CRTO.CRWMT amount of time after the controller is enabled. Refer to Figure 104 for a list of Admin commands that are permitted to be aborted with a status code of Admin Command Media Not Ready.

  The controller shall be able to process without error as described in section 3.5.4.1:

  a) all Admin commands not listed in Figure 104 by the time the controller is ready;
  b) all Admin commands listed in Figure 104 no later than CRTO.CRWMT amount of time after the controller is enabled; and
  c) all NVM commands no later than CRTO.CRWMT amount of time after the controller is enabled.

Figure 104**: Admin Commands Permitted to Return a Status Code of Admin Command Media Not Ready**

Admin Command	Additional Restrictions
Capacity Management	
Device Self-test	If the Device Self-Test would result in testing one or more namespaces, then returning a status code of Admin Command Media Not Ready is permitted. If the Device Self-Test would not result in testing any namespaces, then returning a status code of Admin Command Media Not Ready is not permitted.
Firmware Commit	
Firmware Image Download	
Get LBA Status	
Get Log Page	Get Log Page is only permitted to return a status code of Admin Command Media Not Ready for the following log pages: • Device Self-test • Firmware Slot Information • Telemetry Controller-Initiated • Telemetry Host-Initiated • Predictable Latency Per NVM Set • Predictable Latency Event Aggregate • Persistent Event Log • LBA Status Information • Endurance Group Event Aggregate • Media Unit Status • Supported Capacity Configuration List • Boot Partition • Reservation Notification • Rotational Media Information • Vendor Specific
Namespace Attachment	
Namespace Management	

Figure 104**: Admin Commands Permitted to Return a Status Code of Admin Command Media Not Ready**

Admin Command	Additional Restrictions
Format NVM	
Sanitize	
Security Receive[1]	
Security Send[1]	
Vendor Specific	

Notes:
1. A host may require discovery operations performed via Security Send/Receive (e.g., TCG Level 0 Discovery) to be processed prior to media being ready. Therefore, it is recommended that controllers not return Admin Command Media Not Ready for such discovery operations.

The Controller Ready Modes Supported (CAP.CRMS) field (refer to Figure 36) indicates which controller ready modes are supported. The CAP.CRMS field consists of two bits:

- the Controller Ready With Media Support (CAP.CRMS.CRWMS) bit; and
- the Controller Ready Independent of Media Support (CAP.CRMS.CRIMS) bit.

Controllers shall set the CAP.CRMS.CRWMS bit to '1' (i.e., set the CAP.CRMS field to 01b or 11b). The CAP.CRMS.CRWMS bit was not defined prior to NVM Express Base Specification, Revision 2.0. Controllers compliant with revisions earlier than NVM Express Base Specification, Revision 2.0 may clear the CAP.CRMS.CRWMS field to 00b.

The Controller Ready Independent of Media Enable (CC.CRIME) bit (refer to Figure 46) controls the controller ready mode based on the value of the CAP.CRMS field as follows:

a) If the CAP.CRMS field is cleared to 00b, the controller ready mode is not able to be selected. In this case, the read-only CC.CRIME bit shall be cleared to '0' and should be ignored by host software;
b) If the CAP.CRMS field is set to 01b (i.e., the CAP.CRMS.CRIMS bit is cleared to '0' and the CAP.CRMS.CRWMS bit is set to '1'), then the controller is in Controller Ready With Media mode and the read-only CC.CRIME bit shall be cleared to '0'; and
c) If the CAP.CRMS field is set to 11b, then both controller ready modes are supported, and the host may select the controller ready mode by modifying the value of the CC.CRIME bit. In this situation, the host should set the controller ready mode by writing to the CC.CRIME bit before the controller is enabled (e.g., as part of the initialization sequence of actions described in Section 3.5).

3.5.4 Controller Ready Timeouts During Initialization

The CAP.CRMS field was not defined prior to NVM Express Base Specification, Revision 2.0. Controllers compliant with revisions earlier than NVM Express Base Specification, Revision 2.0 may clear the CAP.CRMS field to 00b. This section is applicable to controllers that clear the CAP.CRMS field to 00b and controllers that set CAP.CRMS to a non-zero value.

There are three controller ready timeout fields:

1. CAP.TO (refer to Figure 36);
2. CRTO.CRWMT (refer to Figure 62); and
3. CRTO.CRIMT (refer to Figure 62).

The details regarding these timeouts during controller initialization are as follows:

a) The CAP.TO field shall be set as described in Figure 36;
b) If the CAP.CRMS field is cleared to 00b', then:
    i. the Controller Ready Independent of Media Timeout (CRTO.CRIMT) field is reserved;
    ii. the Controller Ready With Media Timeout (CRTO.CRWMT) field is reserved; and

iii. the worst-case time the host should wait after the controller is enabled (i.e., CC.EN transitions from ‘0’ to ‘1’) for the controller to become ready (CSTS.RDY transitions from ‘0’ to ‘1’) is indicated by CAP.TO;

c) If the controller is in Controller Ready With Media mode (i.e., the CC.CRIME bit is cleared to ‘0’), then:

i. the Controller Ready Independent of Media Timeout (CRTO.CRIMT) field is not applicable; and
ii. the Controller Ready With Media Timeout (CRTO.CRWMT) indicates the worst-case time the host should wait after the controller is enabled for:

1. the controller to become ready and be able to process all commands without error as described in section 3.5.4.1; and
2. all attached namespaces and media required to process Admin commands to become ready;

and

d) If the controller is in Controller Ready Independent of Media mode (i.e., the CC.CRIME bit is set to ‘1’), then

i. the Controller Ready With Media Timeout (CRTO.CRWMT) field indicates the worst-case time that host software should wait for all attached namespaces and media required to process Admin commands to become ready after the controller is enabled; and
ii. the Controller Ready Independent of Media Timeout (CRTO.CRIMT) field indicates the worst-case time the host should wait after the controller is enabled for the controller to become ready and be able to process:

1. all commands that do not access attached namespaces; and
2. Admin commands that do not require access to media

without error as described in section 3.5.4.1.

Changes to the value of the CC.CRIME bit shall have no effect on the values of the CRTO.CRWMT and CRTO.CRIMT fields. Changes to the value of the CC.CRIME bit may have an effect on the value of the CAP.TO field (refer to Figure 36).

3.5.4.1 Handling Errors During Initialization

If the CAP.CRMS field is non-zero and the controller has been enabled by transitioning CC.EN from ‘0’ to ‘1’ and the controller encounters a failure that prevents:

a) at least one:

- command that does not access attached namespaces; or
- Admin command that does not require access to media (refer to Figure 104),

from being able to be processed without error by the amount of time indicated by the:

- Controller Ready Independent of Media Timeout (CRTO.CRIMT) field since the controller was enabled if the controller is in Controller Ready Independent of Media mode (i.g., the CC.CRIME bit is set to ‘1’); or
- Controller Ready With Media Timeout (CRTO.CRWMT) field since the controller was enabled if the controller is in Controller Ready With Media mode (i.e., the CC.CRIME bit is cleared to ‘0’);

b) at least one namespace attached to the controller from becoming ready by the amount of time indicated by the Controller Ready With Media Timeout (CRTO.CRWMT) field since the controller was enabled; or
c) media required by at least one Admin command from becoming ready by the amount of time indicaated by the Controller Ready With media Timeout (CRTO.CRWMT) field since the controller was enabled,

then:

a) if the controller has not become ready, then the controller shall become ready (i.e., set CSTS.RDY to ‘1’) no later than CRTO.CRWMT amount of time after the controller was enabled; and
b) if the Persistent Event log page is supported, then the controller shall record an NVM Subsystem Hardware Error Event with the NVM Subsystem Hardware Error Event code set to a value of Controller Ready Timeout Exceeded in the Persistent Event log page (refer to Figure 233).

3.6 Shutdown Processing

This section describes the recommended procedure for shutdown processing prior to a power-off condition.

3.6.1 Memory-based Transport Controller Shutdown

It is recommended that the host perform an orderly shutdown of the controller by following the procedure in this section when a power-off or shutdown condition is imminent.

The host should perform the following actions in sequence for a normal controller shutdown:

1. If the controller is enabled (i.e., CC.EN is set to ‘1’):
    a. Stop submitting any new I/O commands to the controller and allow any outstanding commands to complete;
    b. If the controller implements I/O queues, then the host should delete all I/O Submission Queues, using the Delete I/O Submission Queue command. A result of the successful completion of the Delete I/O Submission Queue command is that any remaining commands outstanding are aborted;
    c. If the controller implements I/O queues, then the host should delete all I/O Completion Queues, using the Delete I/O Completion Queue command;

    and
2. The host should set the Shutdown Notification (CC.SHN) field to 01b to indicate a normal controller shutdown operation. The controller indicates when shutdown processing is completed by updating the Shutdown Status (CSTS.SHST) field to 10b and the Shutdown Type (CSTS.ST) field is cleared to ‘0’.

The host should perform the following actions in sequence for an abrupt shutdown:

1. If the controller is enabled (i.e., CC.EN is set to ‘1’), then stop submitting any new I/O commands to the controller; and
2. The host should set the Shutdown Notification (CC.SHN) field to 10b to indicate an abrupt shutdown operation. The controller indicates when shutdown processing is completed by updating the Shutdown Status (CSTS.SHST) field to 10b and CSTS.ST is cleared to ‘0’.

For entry to the D3 power state, the shutdown steps outlined for a normal controller shutdown should be followed.

It is recommended that the host wait a minimum of the RTD3 Entry Latency reported in the Identify Controller data structure for the shutdown operations to complete; if the value reported in RTD3 Entry Latency is 0h, then the host should wait for a minimum of one second. It is not recommended to disable the controller via the CC.EN field. This causes a Controller Reset which may impact the time required to complete shutdown processing. While shutdown processing is in progress, the controller may abort any command with a status code of Commands Aborted due to Power Loss Notification.

It is safe to power off the controller when CSTS.ST is cleared to ‘0’, and CSTS.SHST indicates controller shutdown processing is complete (regardless of the value of CC.EN). It remains safe to power off the controller until CC.EN transitions from ‘0’ to ‘1’.

To start executing commands on the controller after that controller reports controller shutdown processing complete (i.e., CSTS.ST is cleared to ‘0’ and CSTS.SHST is set to 10b) utilizing CC.EN:

- if CC.EN is set to ‘1’, then a Controller Reset (CC.EN cleared from ‘1’ to ‘0’) is required on that controller; or
- if CC.EN is cleared to ‘0’, then the controller is required to be enabled (i.e., CC.EN is set to ‘1’ from ‘0’).

The initialization sequence should then be executed on that controller.

It is an implementation choice whether the host aborts all outstanding commands to the Admin Queue prior to the controller shutdown. The only commands that should be outstanding to the Admin Queue when the controller reports shutdown processing complete are Asynchronous Event Request commands.

3.6.2 Message-based Transport Controller Shutdown

To initiate a shutdown of a controller, the host should use the Property Set command (refer to section 6.6) to set the Shutdown Notification (CC.SHN) field to:

- 01b to initiate a normal shutdown operation; or
- 10b to initiate an abrupt shutdown.

After the host initiates a controller shutdown, the host may either disconnect at the NVMe Transport level or the host may choose to poll CSTS.SHST to determine when the controller shutdown is complete (i.e., the controller should not initiate a disconnect at the NVMe Transport level). It is an implementation choice whether the host aborts all outstanding commands prior to initiating the shutdown.

The CC.EN field is not used to shutdown the controller (i.e., it is used for Controller Reset).

From the time a controller shutdown is initiated until:

- a Controller Level Reset occurs; or
- the controller, if dynamic, is removed from the NVM subsystem,

the controller shall:

- process only Fabrics commands (refer to Figure 375); and
- disable the Keep Alive timer, if supported.

After CC.EN transitions to ‘0’ (i.e., due to Controller Level Reset), the association between the host and controller shall be preserved for at least 2 minutes. After this time, the association may be removed if the controller has not been re-enabled.

3.6.3 NVM Subsystem Shutdown

An NVM Subsystem Shutdown initiates a shutdown of all controllers in a domain or NVM subsystem from a single controller.

3.6.3.1 NVM Subsystem Shutdown in a Single Domain NVM Subsystem

A normal shutdown on all controllers within the NVM subsystem (i.e., normal NVM Subsystem Shutdown) is initiated by:

- a host writing the value 4E726D6Ch ("Nrml") to NSSD.NSSC when CAP.CPS is set to 11b; or
- issuing an NVMe-MI Shutdown command to a Management Endpoint (refer to the NVM Express Management Interface Specification) specifying a normal shutdown.

For each controller in the NVM subsystem for this normal NVM Subsystem Shutdown, if:

- CSTS.SHST is set to 00b; and
- An outstanding Asynchronous Event Request command exists,

then the controller shall issue a Normal NVM Subsystem Shutdown event prior to shutting down the controller.

An abrupt shutdown on all controllers within the NVM subsystem (i.e., abrupt NVM Subsystem Shutdown) is initiated by:

- a host writing the value 41627074h ("Abpt") to NSSD.NSSC when CAP.CPS is set to 11b; or
- issuing an NVMe-MI Shutdown command to a Management Endpoint (refer to the NVM Express Management Interface Specification) specifying an abrupt shutdown.

For either a normal shutdown or an abrupt NVM Subsystem Shutdown, it is safe to power off the NVM subsystem when CSTS.ST is set to '1' and CSTS.SHST indicates shutdown processing complete (i.e., CSTS.SHST is set to 10b) on any controller in the NVM subsystem. It remains safe to power off the NVM subsystem until an NVM Subsystem Reset occurs.

If a normal or abrupt NVM Subsystem Shutdown is being processed or completed within the NVM subsystem (i.e., CSTS.ST is set to '1' and CSTS.SHST is set to 01b or 10b on all controllers in the NVM subsystem), then:

- an NVM Subsystem Reset clears CSTS.SHST to 00b in all controllers in the NVM subsystem; and
- any other type of Controller Level Reset has no effect on the processing of that NVM Subsystem Shutdown.

3.6.3.2 Domain Shutdown in a Multiple Domain NVM Subsystem

A normal NVM Subsystem Shutdown on this controller and all controllers within the associated domain is initiated by:

- a host writing the value 4E726D6Ch ("Nrml") to NSSD.NSSC when CAP.CPS is set to 10b; or
- issuing an NVMe-MI Shutdown command to a Management Endpoint (refer to the NVM Express Management Interface Specification) specifying a normal shutdown.

For each controller in the domain for this normal NVM subsystem shutdown, if:

- CSTS.SHST is cleared to 00b; and
- An outstanding Asynchronous Event Request command exists,

then the controller shall issue a Normal NVM Subsystem Shutdown event prior to shutting down the controller.

An abrupt NVM Subsystem Shutdown to this controller and all controllers within the associated domain is initiated by:

- a host writing the value 41627074h ("Abpt") to NSSD.NSSC when CAP.CPS is set to 10b; or
- issuing an NVMe-MI Shutdown command to a Management Endpoint (refer to the NVM Express Management Interface Specification) specifying an abrupt shutdown.

For either a normal or abrupt NVM Subsystem Shutdown on the domain, it is safe to power off the domain when CSTS.ST is set to '1' and CSTS.SHST indicates shutdown processing complete (i.e., CSTS.SHST is set to 10b) on any controller in the domain. It remains safe to power off the domain until an NVM Subsystem Reset occurs on that domain.

If a normal or abrupt NVM Subsystem Shutdown is being processed or completed within a domain (i.e., CSTS.ST is set to '1' and CSTS.SHST is set to 01b or 10b on all controllers in the domain), then:

- an NVM Subsystem Reset clears CSTS.SHST to 00b in all controllers in the Domain; and
- any other type of Controller Level Reset has no effect on the processing of that shutdown.

3.7 Resets

3.7.1 NVM Subsystem Reset

3.7.1.1 Single Domain NVM Subsystems

The scope of an NVM Subsystem Reset depends on whether the NVM subsystem supports multiple domains. In an NVM subsystem that does not support multiple domains, the scope of the NVM Subsystem Reset is the entire NVM subsystem.

An NVM Subsystem Reset is initiated when:

- Main power is applied to the NVM subsystem;
- A value of 4E564D65h (“NVMe”) is written to the NSSR.NSSRC field;
- Requested using a method defined in the NVM Express Management Interface Specification; or
- A vendor specific event occurs.

When an NVM Subsystem Reset occurs, the entire NVM subsystem is reset. This includes the initiation of a Controller Level Reset on all controllers that make up the NVM subsystem, disabling of the Persistent Memory Region associated with all controllers that make up the NVM subsystem, and any transport specific actions defined in the applicable NVMe transport specification.

The occurrence of an NVM Subsystem Reset while power is applied to the NVM subsystem is reported by the initial value of the CSTS.NSSRO field following the NVM Subsystem Reset. This field may be used by host software to determine if the sudden loss of communication with a controller was due to an NVM Subsystem Reset or some other condition.

The ability for host software to initiate an NVM Subsystem Reset by writing to the NSSR.NSSRC field is an optional capability of a controller indicated by the state of the CAP.NSSRS field. An implementation may protect the NVM subsystem from an inadvertent NVM Subsystem Reset by not providing this capability to one or more controllers that make up the NVM subsystem.

The occurrence of a vendor specific event that results in an NVM Subsystem Reset is intended to allow implementations to recover from a severe NVM subsystem internal error that prevents continued normal operation (e.g., fatal hardware or firmware error).

3.7.1.2 Multiple Domain NVM Subsystems

The scope of an NVM Subsystem Reset depends on whether the NVM subsystem supports multiple domains. In an NVM subsystem that supports multiple domains, the scope of the NVM Subsystem Reset is either the controllers that are in a domain or the entire NVM subsystem.

An NVM Subsystem Reset on a domain is initiated when:

- Power is applied to that domain;
- A value of 4E564D65h (i.e., “NVMe”) is written to the NSSR.NSSRC field of one of the controllers in that domain; or
- A vendor specific event occurs within that domain.

When an NVM Subsystem Reset occurs the entire domain is reset. This includes the initiation of a Controller Level Reset on all controllers that are in the domain, disabling of the Persistent Memory Region associated with all controllers that are in the domain, and any transport specific actions defined in the applicable NVMe transport specification.

Alternatively, an NVM Subsystem Reset in an NVM subsystem that supports multiple domains may reset the entire NVM subsystem.

The occurrence of an NVM Subsystem Reset while power is applied to the domain is reported by the initial value of the CSTS.NSSRO field following the NVM Subsystem Reset. This field may be used by host software to determine if the sudden loss of communication with a controller was due to an NVM Subsystem Reset or some other condition.

The ability for host software to initiate an NVM Subsystem Reset by writing to the NSSR.NSSRC field is an optional capability of a controller indicated by the state of the CAP.NSSRS field. An implementation may protect the domain from an inadvertent NVM Subsystem Reset by not providing this capability to one or more controllers that are in the domain.

3.7.2 Controller Level Reset

The following methods initiate a Controller Level Reset:

- NVM Subsystem Reset;

- Controller Reset (i.e., CC.EN transitions from ‘1’ to ‘0’); and
- Transport specific reset types (refer to the applicable NVMe Transport binding specification), if any.

A Controller Level Reset consists of the following actions:

- The controller stops processing any outstanding Admin or I/O commands;
- All I/O Submission Queues are deleted;
- All I/O Completion Queues are deleted;
- The controller is brought to an Idle state. When this is complete, CSTS.RDY is cleared to ‘0’; and
- All controller properties defined in section 3.1.3 and internal controller state are reset, with the following exceptions:
    - for controllers using a memory-based transport:
        - the Admin Queue properties (AQA, ASQ, or ACQ) are not reset as part of a Controller Reset;
        - the Controller Memory Buffer Memory Space Control property (CMBMSC) is reset as part of neither a Controller Reset nor a Function Level Reset; and
        - the Persistent Memory Region Memory Space Control Upper property (PMRMSCU) and the Persistent Memory Region Memory Space Control Lower property (PMRMSCL) are not reset as part of a Controller Reset;

      and
    - for controllers using a message-based transport:
        - there are no exceptions.

In all Controller Level Reset cases except a Controller Reset, the controller properties are reset as defined by the applicable NVMe Transport binding specification.

To continue after a Controller Level Reset, the host shall:

- Update transport specific state and controller property state as appropriate;
- Set CC.EN to ‘1’;
- Wait for CSTS.RDY to be set to ‘1’;
- Configure the controller using Admin commands as needed;
- Create I/O Completion Queues and I/O Submission Queues as needed; and
- Proceed with normal I/O operations.

Note that all Controller Level Reset cases except a Controller Reset result in the controller immediately losing communication with the host. In all these cases, the controller is unable to indicate any aborts or update any completion queue entries.

3.7.3 Queue Level Reset

The host may reset and/or reconfigure the I/O Submission and I/O Completion Queues by resetting them. A queue level reset is performed by deleting and then recreating the queue. In this process, the host should wait for all pending commands to the appropriate I/O Submission Queue(s) to complete.

To perform the queue level reset on a controller using the memory-based transport model, the host submits the Delete I/O Submission Queue or Delete I/O Completion Queue command to the Admin Queue specifying the identifier of the queue to be deleted. After successful command completion of the queue delete operation, the host then recreates the queue by submitting the Create I/O Submission Queue or Create I/O Completion Queue command. As part of the creation operation, the host may modify the attributes of the queue. Note that if a queue level reset is performed on an I/O Completion Queue, the I/O Submission Queues that are utilizing the I/O Completion Queue should be deleted before the I/O Completion Queue is reset and recreated after the I/O Completion Queue is recreated. The behavior of an I/O Submission Queue without a corresponding I/O Completion Queue is undefined.

To perform the queue level reset on a controller using the message-based transport model, the host sends a Disconnect command to the I/O Queue which is to be deleted. After successful command completion of the Disconnect command, the host then recreates the I/O Submission Queue and I/O Completion Queue by submitting the Connect command with a QID specified that is not 00h. As part of the Connect command, the host may modify the attributes of the I/O queues.

The host should ensure that the appropriate I/O Submission Queue or I/O Completion Queue is idle before deleting that queue. Submitting a queue deletion command causes any pending commands to be aborted by the controller; this may or may not result in a completion queue entry being posted for the aborted command(s).

3.8 NVM Capacity Model

3.8.1 Overview

NVM subsystems may report capacity-related information for multiple entities within the NVM subsystem. This capacity reporting model includes capacity reporting for the NVM subsystem, the domain (refer to section 3.2.4), the Endurance Group (refer to section 3.2.3), the NVM Set (refer to section 3.2.2), the namespace (refer to section 3.2.1), and the Media Unit (refer to section 1.5.34). Some, all, or none of this reporting may be supported by an NVM subsystem.

Figure 12 shows the hierarchical relationships of the entities within an NVM subsystem which are used to manage NVM capacity.

The capacity in an NVM Set is able to be allocated to one or more namespaces, and each namespace exists entirely in that NVM Set (refer to section 3.2.2). Not all of the capacity in the NVM Set is required to be allocated to a namespace.

If the controller supports NVM Sets, then the capacity in an Endurance Group is able to be allocated to one or more NVM Sets and each NVM Set exists entirely in that Endurance Group (refer to section 3.2.3). Not all of the capacity in an Endurance Group is required to be allocated to an NVM Set.

If the controller supports Endurance Groups and does not indicate support for NVM Sets, then in all data structures that contain an NVMSETID field, the NVMSETID field shall be cleared to 0h.

If the controller does not support Endurance Groups, then in all data structures that contain an ENDGID field, the ENDGID field shall be cleared to 0h.

If the controller supports Endurance Groups, then the capacity in a domain is able to be allocated to one or more Endurance Groups, and each Endurance Group exists entirely in that domain (refer to section 3.2.4). Not all of the capacity in a domain is required to be allocated to an Endurance Group.

NVM subsystems may report the composition of Endurance Groups and NVM Sets in terms of Media Units. Each Media Unit is allocated to exactly one Endurance Group. If NVM Sets are supported, each Media Unit is allocated to exactly one NVM Set. Data is transferred to and from Media Units via Channels. Each Media Unit is connected to one or more Channels. Each Channel is connected to one or more Media Units.

A host uses Capacity Management (refer to section 8.3) to allocate:

a) Domain capacity to Endurance Groups;
b) Endurance Group capacity to NVM Sets;
c) Media Units to Endurance Groups; and
d) Media Units to NVM Sets,

as part of creating those entities.

A host uses the Namespace Management create operation (refer to section 8.11) to allocate capacity to namespaces.

3.8.2 Media Unit Organization Examples

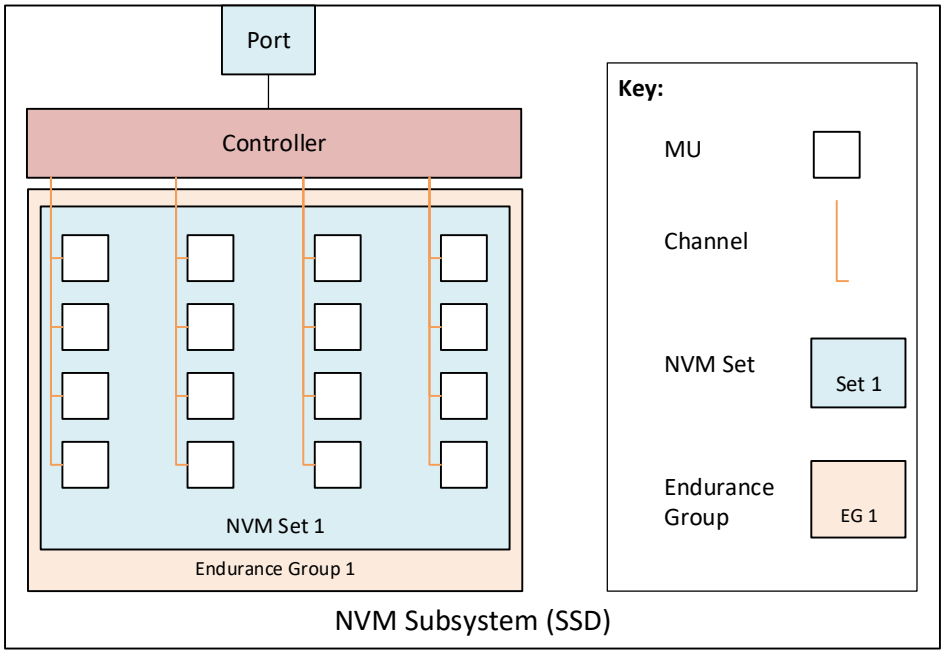
Allocation of Media Units is used to organize the physical NVM resources in an NVM subsystem to meet particular performance goals.

The following examples show an NVM subsystem with all resources in a single domain. The domain has four Channels, with four Media Units attached to each Channel.

3.8.2.1 Simple NVM Subsystem

Figure 105 shows an example of a single domain NVM subsystem where endurance is managed across all media units. The performance goal is maximum bandwidth, which is achieved by allowing each read or write operation to simultaneously access all Media Units. All Media Units are in the same Endurance Group and in the same NVM Set.

Figure 105: Simple NVM Subsystem

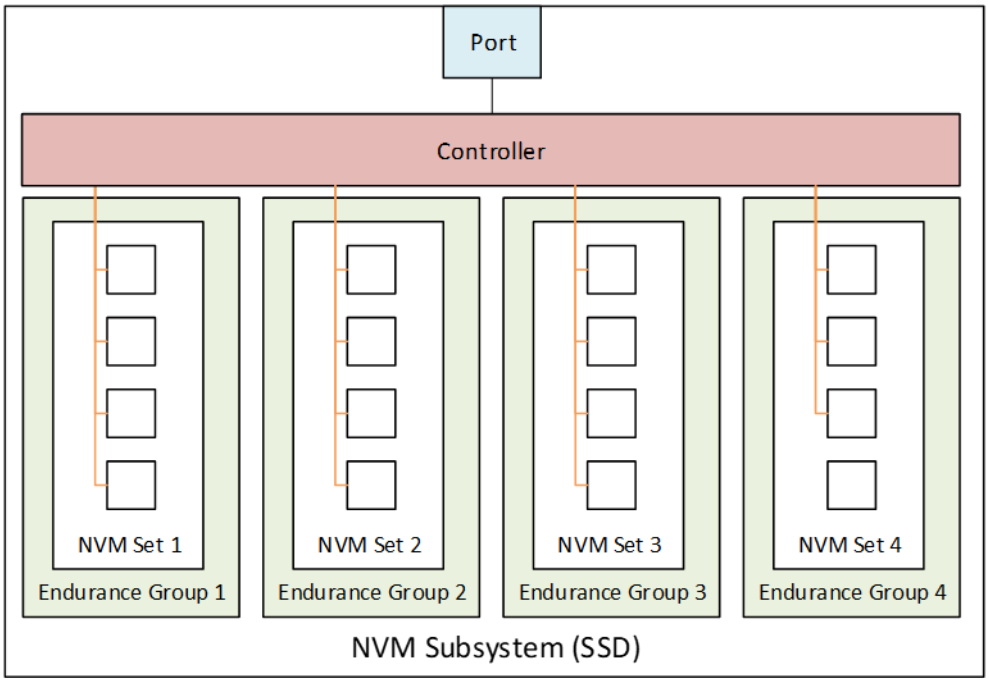


The Capacity Configuration Descriptor for this example contains one Endurance Group Configuration Descriptor. The Endurance Group Configuration Descriptor contains one NVM Set Identifier and four Channel Configuration Descriptors. Each Channel Configuration Descriptor contains four Media Unit Configuration Descriptors.

3.8.2.2 Vertically-Organized NVM Subsystem

Figure 106 shows an example of a single domain NVM subsystem where the performance goal is isolation among four NVM Sets at the cost of bandwidth. Endurance is managed separately for each set. Media Units sharing a Channel are allocated to the same Endurance Group. All Media Units in an Endurance Group are allocated to the same NVM Set. The bandwidth for any NVM Set is likely to be less than or equal to the bandwidth of the Channel of that NVM Set.

Figure 106: Vertically-Organized NVM Subsystem

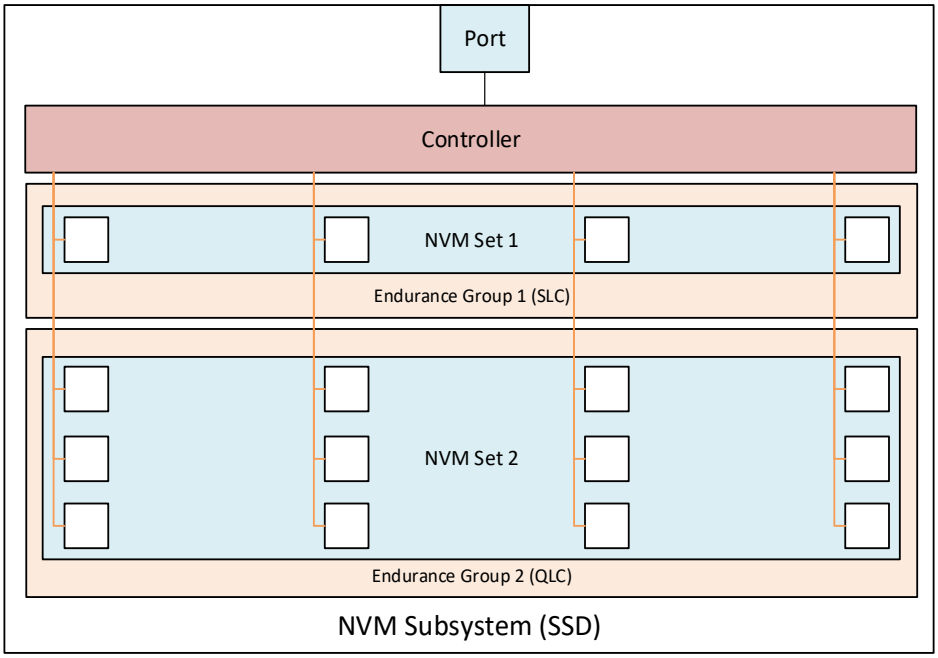


The Capacity Configuration Descriptor for this example contains four Endurance Group Configuration Descriptors. Each Endurance Group Configuration Descriptor contains one NVM Set Identifier and one Channel Configuration Descriptor. Each Channel Configuration Descriptor contains four Media Unit Configuration Descriptors.

3.8.2.3 Horizontally-Organized Dual NAND NVM Subsystem

Figure 107 shows an example of a single domain NVM subsystem where the Media Units are NAND which is capable of being operated as QLC or at a lower density. The performance goal is for maximum bandwidth to a small NVM Set operating as SLC and to a larger NVM Set operating as QLC.

Figure 107: Horizontally-Organized Dual NAND NVM Subsystem



The Capacity Configuration Descriptor, for this example, contains two Endurance Group Configuration Descriptors. The first Endurance Group Configuration Descriptor for this example:

- indicates a Capacity Adjustment Factor of approximately 400;
- contains one NVM Set Identifier; and
- contains four Channel Configuration Descriptors. Each Channel Configuration Descriptor contains one Media Unit Configuration Descriptor.

The second Endurance Group Configuration Descriptor for this example:

- indicates a Capacity Adjustment Factor of 100;
- contains one NVM Set Identifier; and
- contains four Channel Configuration Descriptors. Each Channel Configuration Descriptor contains three Media Unit Configuration Descriptors.

3.8.3 Capacity Reporting

For an NVM subsystem that does not support multiple domains, the capacity information reported in the Identify Controller data structure (i.e., the TNVMCAP field and the UNVMCAP field in Figure 275) describes the capacity for the NVM subsystem. If the MEGCAP field is non-zero, that field indicates the largest entity (e.g., Endurance Group, NVM Set, namespace) that may be created in the NVM subsystem.

For an NVM subsystem that supports multiple domains, the capacity information reported in the Identify Controller data structure describes the capacity accessible by the controller processing the Identify command. The host may use the Identify command to access the Domain List data structure (refer to section 5.17.2.17) to determine the domains that are accessible by the controller and the capacity information for each of those domains. If the Max Endurance Group Domain Capacity field is non-zero, then the field describes the largest entity (e.g., Endurance Group, namespace) that may be created by this controller in the domain described by that Domain Attributes Entry.

For an NVM subsystem that supports Endurance Groups (refer to section 3.2.3), the host may use the Identify command to access the Endurance Group List data structure (refer to section 5.17.2.18) to determine the Endurance Groups that are accessible by the controller. To determine the capacity information for each Endurance Group, the host uses the Get Log Page command to access the Endurance Group Information log page (refer to section 5.16.1.10).

For an NVM subsystem that supports NVM Sets (refer to section 3.2.2), the host may use the Identify command to access the NVM Set List data structure (refer to section 5.17.2.4) to determine the NVM Sets that are accessible by the controller and the capacity information for each of those NVM Sets.

For the management of Endurance Groups, NVM Sets, and namespaces, Figure 108 describes the effects of the support of NVM Sets, Endurance Groups, and domains on which capacity information is used for each management operation.

Figure 108: Capacity Information Field Usage

Entity being created / deleted	NVM Sets supported	Endurance Groups supported	Domains supported	Capacity information used [1]
Endurance Group [7]	n/a	Yes	No	NVM subsystem [3]
			Yes	Domain [4]
NVM Set [7]	Yes	Yes [2]	n/a	Endurance Group [5]
Namespace [8]	No	No	No	NVM subsystem [3]
			Yes	Domain [4]
		Yes	n/a	Endurance Group [5]
	Yes	Yes [2]	n/a	NVM Set [6]

Notes:
1. This information described in this column is used by the host for creating the entity (e.g., to determine if there is sufficient available capacity) and this information is altered by the controller as a result of the creation or deletion of the entity (e.g., unallocated capacity decreased as a result of entity creation, or unallocated capacity increased as a result of entity deletion).
2. NVM Set support requires support for Endurance Groups as described in section 3.2.2.
3. Capacity information in the Identify Controller data structure (i.e., TNVMCAP field, UNVMCAP field, and MEGCAP fields (refer to Figure 275)).
4. Capacity information in the Domain Attributes Entry (i.e., Total Domain Capacity field, Unallocated Domain Capacity field, and Max Endurance Group Domain Capacity field (refer to Figure 287)).
5. Capacity information in the Endurance Group Information log page (i.e., TEGCAP field, UEGCAP field (refer to Figure 217)).
6. Capacity information in the NVM Set Attributes Entry (i.e., Total NVM Set Capacity field, and Unallocated NVM Set Capacity field (refer to Figure 279)).
7. Endurance Groups and NVM Sets are created and deleted using the Capacity Management command (refer to section 5.3)
8. Namespaces are created and deleted using the Namespace Management command (refer to section 8.11). Namespaces are deleted using the Capacity Management command.

3.9 Keep Alive

The Keep Alive timer is a watchdog timer intended to detect a malfunctioning connection, controller, or host. The Keep Alive Timeout Interval is the period during which the Keep Alive Timer is activated.

A Keep Alive Timeout Interval on the controller starts when:

- a successful completion queue entry is posted for a Set Features command with Feature Identifier 0Fh and a non-zero KATO field.

A Keep Alive Timeout Interval on the host starts when:

- a Set Features command with Feature Identifier 0Fh and a non-zero Keep Alive Timeout (KATO) field is posted to the Admin submission queue; or
- a Keep Alive command was posted to the Admin submission queue.

Both on the host and the controller the Keep Alive Timeout Interval ends the time specified by the KATO field after the interval started (refer to Figure 341). A Keep Alive Timeout occurs when the Keep Alive Timer expires. The Keep Alive Timer expires:

- if the TBKAS bit is cleared to '0', at the end of the Keep Alive Timeout Interval and no Keep Alive Command was processed during the Keep Alive Timeout Interval (refer to section 3.9.2); and
- if the TBKAS bit is set to '1', at the end of the Keep Alive Timeout Interval and no Admin command or I/O command was processed during the Keep Alive Timeout Interval (refer to section 3.9.2).

The Keep Alive Timeout is the maximum time a connection remains established without processing a Keep Alive command. The Keep Alive timer in the controller expires when a Keep Alive command is not received within the Keep Alive Timeout interval.

The Keep Alive timer is active if:

- CC.EN is set to '1';
- CSTS.RDY is set to '1';
- CC.SHN is cleared to '00b';
- CSTS.SHST is cleared to '00b'; and
- the Keep Alive Timer feature has been enabled with a KATO field (refer to section 5.27.1.12 and section 6.3) set to a non-zero value,

otherwise, the Keep Alive timer is inactive and a Keep Alive Timeout shall not occur. Activating an inactive Keep Alive timer (e.g., enabling a controller with the Keep Alive feature in use, enabling a controller that supports NVMe over Fabrics where the Connect command specified a non-zero Keep Alive Timeout (refer to section 3.1.2.3)) shall initialize the Keep Alive timer to the Keep Alive Timeout value.

The host may consider a Keep Alive Timeout to have occurred when the completion of the Keep Alive command is not received within the Keep Alive Timeout interval. The host is intended to send Keep Alive commands at a faster rate than the Keep Alive Timeout accounting for transport roundtrip times, transport delays, command execution times, and the Keep Alive Timer granularity.

If a Keep Alive Timer expires:

a) the controller shall:

   - record an Error Information Log Entry with the status code Keep Alive Timeout Expired,
   - stop processing commands;
   - set the Controller Fatal Status (CSTS.CFS) bit to '1'; and
   - for message-based NVMe Transports:
     - terminate the NVMe Transport connection; and
     - break the host to controller association;

   and

b) the host assumes all outstanding commands are not completed and re-issues commands as appropriate.

For message-based NVMe Transports, after completing these steps, a controller may accept a Connect command (refer to section 6.3) for the Admin Queue from the same or another host in order to form a new association.

The Keep Alive command restarts the timeout period; other commands have no effect on the timeout. The controller should process the Keep Alive command as soon as the command is fetched.

The NVMe Transport binding specification defines for the associated NVMe Transport:

- the minimum Keep Alive Timeout value;
- the maximum Keep Alive Timeout value; and
- if support for the Keep Alive feature is required.

NVMe Transports that do not detect a connection loss in a timely manner shall require that the Keep Alive feature be enabled. If a command attempts to disable the Keep Alive timer by setting the Keep Alive Timeout

value to 0h or to a value that exceeds the maximum allowed by the associated NVMe Transport binding specification, a status code of Keep Alive Timeout Invalid shall be returned. If a command sets the Keep Alive Timeout value to a value that is less than the minimum supported by the NVMe Transport or less than the minimum supported by the specific implementation, then the controller sets the Keep Alive Timeout value to that minimum value.

3.9.1 Keep Alive Command Based Keep Alive

Keep Alive Command Based Keep Alive restricts the Keep Alive Timer on both the host and the controller to be restarted only upon the processing of a Keep Alive command. This mode is in use if the TBKAS bit is cleared to '0'.

The Keep Alive Timeout is the maximum time a connection remains established without processing a Keep Alive command. If the Keep Alive Timer in the controller expires and a Keep Alive command has not been processed within the Keep Alive Timeout Interval, then the controller may consider a Keep Alive Timeout to have occurred. If the Keep Alive Timer in the host expires and a completion of a Keep Alive command has not been received with the Keep Alive Timeout Interval, then the host may consider a Keep Alive Timeout to have occurred. The host should send Keep Alive commands at half of the Keep Alive Timeout accounting for transport roundtrip times, transport delays, command processing times, and the Keep Alive Timer granularity.

3.9.2 Traffic Based Keep Alive

Traffic Based Keep Alive (TBKAS) allows the host and controller to restart the Traffic Based Keep Alive Timer in the presence of Admin or I/O command processing. The controller support for the TBKAS bit is indicated in the Controller Attributes in the Identify Controller data structure (refer to Figure 275). If the Controller does not support Traffic Based Keep Alive (TBKAS is cleared to '0'), then the operation of the Keep Alive feature is described in section 3.9.1.

The Traffic Based Keep Alive Timeout occurs if a connection remains established without processing an Admin command or an I/O command during the Keep Alive Timeout Interval. If an Admin command or an I/O command is processed within the Keep Alive Timeout Interval, then upon the expiration of the Keep Alive Timer, the Keep Alive Timer shall be restarted.

The controller may consider a Keep Alive Timeout to have occurred if no Admin command or no I/O command is submitted to the controller (as defined in section 3.4.4) within the Keep Alive Timeout Interval. If an Admin command or an I/O command is transferred to the Controller within the Keep Alive Timeout Interval, then upon the expiration of the Keep Alive Timer the controller shall restart the Keep Alive Timer.

The host may consider a Traffic Based Keep Alive Timeout to have occurred if the host does not receive a completion of any Admin command or any I/O command within the Keep Alive Timeout Interval. If an Admin command or an I/O command is completed within the Keep Alive Timeout Interval, then upon expiration of the Keep Alive Timer, the host shall restart the Keep Alive Timer. The host should check for a command completion queue entry for any Admin commands and I/O commands at half of the Keep Alive Timeout accounting for transport roundtrip times, transport delays, command processing times, and the Keep Alive Timer granularity. To prevent the controller from detecting a Keep Alive Timeout, if no Admin command and no I/O command is sent to the controller during half of the Keep Alive Timeout Interval, the host should send a Keep Alive command.

3.10 Privileged Actions

Privileged actions are actions (e.g., command, property write) that affect or have the potential to affect the state beyond the controller and attached namespaces.

Examples of privileged actions are:

- Admin commands including Namespace Management, Namespace Attachment, Virtualization Management, Format NVM, Set Features with Feature Identifier 17h (i.e., Sanitize Config, refer to section 5.27.1.19), Sanitize, and Capacity Management;
- Property Writes including NVM Subsystem Reset; and

- Some Vendor specific commands and properties.

3.11 Firmware Update Process

The process for a firmware update to be activated in a domain (refer to section 3.2.4) by a reset is:

1. The host issues a Firmware Image Download command to download the firmware image to a controller. There may be multiple portions of the firmware image to download, thus the offset for each portion of the firmware image being downloaded on that controller is specified in the Firmware Image Download command. The data provided in the Firmware Image Download command should conform to the Firmware Update Granularity indicated in the Identify Controller data structure or the firmware update may fail;
2. After the firmware is downloaded to that controller, the next step is for the host to submit a Firmware Commit command to that controller. The Firmware Commit command verifies that the last firmware image downloaded is valid and commits that firmware image to the firmware slot indicated for future use. A firmware image that does not start at offset zero, contains gaps, or contains overlapping regions is considered invalid. A controller may employ additional vendor specific means (e.g., checksum, CRC, cryptographic hash, or a digital signature) to determine the validity of a firmware image:

    a. The Firmware Commit command may also be used to activate a firmware image associated with a previously committed firmware slot;

3. The last step is to perform a reset that then causes the firmware image specified in the Firmware Slot field in the Firmware Commit command to be activated. The reset may be an NVM Subsystem Reset, Conventional Reset, Function Level Reset, or Controller Reset (CC.EN transitions from '1' to '0'):

    a. In some cases a Conventional Reset or NVM Subsystem Reset is required to activate a firmware image. This requirement is indicated by Firmware Commit command specific status (refer to section 5.12.1);

    and

4. After the reset has completed, host software re-initializes the controller. This includes re-allocating I/O Submission and Completion Queues. Refer to sections 3.5.1 and 3.5.2.

The process for a firmware update to be activated on a domain without a reset is:

1. The host issues a Firmware Image Download command to download the firmware image to a controller. There may be multiple portions of the firmware image to download, thus the offset for each portion of the firmware image being downloaded on that controller is specified in the Firmware Image Download command. The data provided in the Firmware Image Download command should conform to the Firmware Update Granularity indicated in the Identify Controller data structure or the firmware update may fail;
2. The host submits a Firmware Commit command on that controller with a Commit Action of 011b which specifies that the firmware image should be activated immediately without reset. The downloaded firmware image should replace the firmware image in the firmware slot. If no firmware image was downloaded since the last reset or Firmware Commit command, (i.e., the first step was skipped), then that controller shall verify and activate the firmware image in the specified slot. If that controller starts to activate the firmware image, any controllers affected by the new firmware image send a Firmware Activation Starting asynchronous event to the host if Firmware Activation Notices are enabled (refer to Figure 326):

    a. The Firmware Commit command may also be used to activate a firmware image associated with a previously committed firmware slot;

3. The controller completes the Firmware Commit command. The following actions are taken in certain error scenarios:

a. If the firmware image is invalid, then the controller reports the appropriate error (e.g., Invalid Firmware Image);
b. If the firmware activation was not successful because a Controller Level Reset is required to activate this firmware, then the controller reports an error of Firmware Activation Requires Controller Level Reset and the firmware image is applied at the next Controller Level Reset;
c. If the firmware activation was not successful because an NVM Subsystem Reset is required to activate this firmware image, then the controller reports an error of Firmware Activation Requires NVM Subsystem Reset and the image is applied at the next NVM Subsystem Reset;
d. If the firmware activation was not successful because a Conventional Reset is required to activate this firmware, then the controller reports an error of Firmware Activation Requires Conventional Reset and the firmware image is applied at the next Conventional Reset; and
e. If the firmware activation was not successful because the firmware activation time would exceed the MTFA value reported in the Identify Controller data structure, then the controller reports an error of Firmware Activation Requires Maximum Time Violation. In this case, to activate the firmware, the Firmware Commit command needs to be re-issued and the firmware image activated using a reset.

If the controller transitions to the $D3_{cold}$ state (refer to the PCI Express Base Specification) after the submission of a Firmware Commit command that attempts to activate a firmware image and before the completion of that command, then the controller may resume operation with either the firmware image active at the time the Firmware Commit command was submitted or the firmware image that was activated by that command.

If the firmware image is not able to be successfully loaded, then the controller shall revert to the firmware image present in the most recently activated firmware slot or the baseline read-only firmware image, if available, and indicate the failure as an asynchronous event with a Firmware Image Load Error.

If a host overwrites (i.e., updates) the firmware image in the active firmware slot, then the previously active firmware image may no longer be available. As a result, any action (e.g., power cycling the controller) that requires the use of that firmware slot may instead use the firmware image that is currently in that firmware slot.

Host software should not overlap firmware/boot partition image update command sequences (refer to section 1.5.23). During a firmware image update command sequence, if a Firmware Image Download command or a Firmware Commit command is submitted for another firmware/boot partition image update command sequence, the results of both that command and the in-progress firmware image update are undefined.

Host software should use the same controller or Management Endpoint (refer to the NVM Express Management Interface Specification) for all commands that are part of a firmware image update command sequence. If the commands for a single firmware/boot partition image update command sequence are submitted to more than one controller and/or Management Endpoint, the controller may abort the Firmware Commit command with Invalid Firmware Image status.

After downloading a firmware image, host software issues a Firmware Commit command before downloading additional firmware images. Processing of the first Firmware Image Download command after completion of a Firmware Commit command shall cause the controller to discard remaining portions, if any, of downloaded images. If a reset occurs between a firmware download and completion of the Firmware Commit command, then the controller shall discard all portion(s), if any, of downloaded images.

4 Data Structures

This section describes data structures used by the NVM Express.

4.1 Data Layout

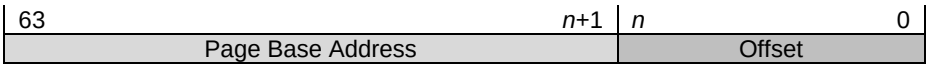
This section describes the data structures used to describe the layout of data that can be understood by the controller and the host.

4.1.1 Physical Region Page Entry and List

A physical region page (PRP) entry is a pointer to a physical memory page. PRPs are used as a scatter/gather mechanism for data transfers between the controller and memory. To enable efficient out of order data transfers between the controller and the host, PRP entries are a fixed size.

The size of the physical memory page is configured by host software in CC.MPS. Figure 109 shows the layout of a PRP entry that consists of a Page Base Address and an Offset. The size of the Offset field is determined by the physical memory page size configured in CC.MPS.

Figure 109: PRP Entry Layout



The definition of a PRP entry is described in Figure 110.

Figure 110: PRP Entry – Page Base Address and Offset

Bits	Description
63:00	**Page Base Address and Offset (PBAO):** This field indicates the 64-bit physical memory page address. The least significant bits (*n*:0) of this field indicate the offset within the memory page (e.g., if the memory page size is 4 KiB, then bits 11:00 form the Offset; if the memory page size is 8 KiB, then bits 12:00 form the Offset). If this entry is not the first PRP entry in the command or a PRP List pointer in a command, then the Offset portion of this field shall be cleared to 0h. The Offset shall be dword aligned, indicated by bits 1:0 being cleared to 00b. Note: The controller is not required to check that bits 1:0 are cleared to 00b. The controller may report an error of PRP Offset Invalid if bits 1:0 are not cleared to 00b. If the controller does not report an error of PRP Offset Invalid, then the controller shall operate as if bits 1:0 are cleared to 00b.

A physical region page list (PRP List) is a set of PRP entries in a single page of contiguous memory. A PRP List describes additional PRP entries that could not be described within the command itself. Any PRP entries described within the command are not duplicated in a PRP List. If the amount of data to transfer requires multiple PRP List memory pages, then the last PRP entry before the end of the memory page shall be a pointer to the next PRP List, indicating the next segment of the PRP List. Figure 111 shows the layout of a PRP List where each PRP entry identifies memory pages that are physically contiguous. Figure 112 shows the layout of a PRP List where each PRP entry identifies a different memory page (i.e., the memory pages are not physically contiguous).

Figure 111: PRP List Layout for Physically Contiguous Memory Pages

63 **n+1**	***n*** **0**
Page Base Address *p*	0h
Page Base Address *p*+1	0h
…	
Page Base Address *p*+*q*	0h
Page Base Address *p*+*q*+1	0h

Figure 112: PRP List Layout for Physically Non-Contiguous Memory Pages

63 *n*+1	*n* 0
Page Base Address p	0h
Page Base Address q	0h
…	
Page Base Address r	0h
Page Base Address s	0h

Dependent on the command definition, the first PRP entry contained within the command may have a non-zero offset within the memory page. The first PRP List entry (i.e., the first pointer to a memory page containing additional PRP entries) that if present is typically contained in the PRP Entry 2 location within the command, shall be qword aligned and may also have a non-zero offset within the memory page.

PRP entries contained within a PRP List shall have a memory page offset of 0h. If a second PRP entry is present within a command, it shall have a memory page offset of 0h. In both cases, the entries are memory page aligned based on the value in CC.MPS. If the controller receives a non-zero offset for these PRP entries the controller should return an error of PRP Offset Invalid.

PRP Lists shall be minimally sized with packed entries starting with entry 0. If more PRP List pages are required, then the last entry of the PRP List contains the Page Base Address of the next PRP List page. The next PRP List page shall be memory page aligned. The total number of PRP entries required by a command is implied by the command parameters and memory page size.

4.1.2 Scatter Gather List (SGL)

A Scatter Gather List (SGL) is a data structure in memory address space used to describe a data buffer. The controller indicates the SGL types that the controller supports in the Identify Controller data structure. A data buffer is either a source buffer or a destination buffer. An SGL contains one or more SGL segments. The total length of the Data Block and Bit Bucket descriptors in an SGL shall be equal to or exceed the amount of data requested to be transferred. If the amount of data requested to be transferred exceeds the total length of the Data Block and Bit Bucket descriptors in an SGL, data shall not be transferred to or from locations that are not described by the SGL.

An SGL segment is a qword aligned data structure in a contiguous region of physical memory describing all, part of, or none of a data buffer and the next SGL segment, if any. An SGL segment consists of an array of one or more SGL descriptors. Only the last descriptor in an SGL segment may be an SGL Segment descriptor or an SGL Last Segment descriptor.

A last SGL segment is an SGL segment that does not contain an SGL Segment descriptor, or an SGL Last Segment descriptor.

A controller may support byte or dword alignment and granularity of Data Blocks. If a controller supports only dword alignment and granularity as indicated in the SGL Support field of the Identify Controller data structure (refer to Figure 275), then the values in the Address and Length fields of all Data Block descriptors shall have their two least significant bits cleared to 00b. This requirement applies to Data Block descriptors that indicate data and/or metadata memory regions.

The SGL Descriptor Threshold (SDT) field in the Identify Controller data structure (refer to Figure 275) indicates the recommended maximum number of SGL descriptors for a command. If the SDT field is set to a non-zero value, and a command is submitted for which the sum of:

a) the number of SGL Bit Bucket descriptors with non-zero Length field contents; and
b) the number of SGL Data Block descriptors with a non-zero Length field contents,

exceeds the value of the SDT field, then the performance of the controller may be reduced.

The value of the SDT field shall be less than or equal to the value of the Maximum SGL Data Block Descriptors (MSDBD) field in the Identify Controller data structure (refer to Figure 275 for the definition of the MSDBD field).

A Keyed SGL Data Block descriptor is a Data Block descriptor that includes a key that is used as part of the host memory access. The maximum length that may be specified in a Keyed SGL Data Block descriptor is (16 MiB – 1).

A Transport SGL Data Block descriptor is a Data Block descriptor that specifies a data block that is transferred by the NVMe Transport using a transfer mechanism and data buffers that are specific to the NVMe Transport.

The SGL Identifier Descriptor Sub Type field may indicate additional information about a descriptor. As an example, the Sub Type may indicate that the Address field is an offset rather than an absolute address. The Sub Type may also indicate NVMe Transport specific information.

The controller shall abort a command if:

- an SGL segment contains an SGL Segment descriptor or an SGL Last Segment descriptor in other than the last descriptor in the segment;
- a last SGL segment contains an SGL Segment descriptor, or an SGL Last Segment descriptor;
- an SGL descriptor has an unsupported format; or
- an SGL Data Block descriptor contains Address or Length fields with either of the two least significant bits set to 1b and the controller supports only dword alignment and granularity as indicated in the SGL Support field of the Identify Controller data structure. Refer to Figure 275.

Figure 113 defines the SGL segment.

Figure 113: SGL Segment

Bytes	Description
15:00	**SGL Descriptor 0**
31:16	**SGL Descriptor 1**
…	…
((n*16)+15):(n*16)	**SGL Descriptor n**

An SGL segment contains one or more SGL descriptors. Figure 114 defines the generic SGL descriptor format.

Figure 114: Generic SGL Descriptor Format

Bytes	Description
14:00	**Descriptor Type Specific**
15	**SGL Identifier:** The definition of this field is described in the table below. **Bits** / **Description** 03:00 – SGL Descriptor Sub Type (refer to Figure 116) 07:04 – SGL Descriptor Type (refer to Figure 115)

The SGL Descriptor Type field defined in Figure 115 specifies the SGL descriptor type. If the SGL Descriptor Type field is set to a reserved value or an unsupported value, then the SGL descriptor shall be processed as having an SGL Descriptor Type error. If the SGL Descriptor Sub Type field is set to a reserved value or an unsupported value, then the descriptor shall be processed as having an SGL Descriptor Type error.

An SGL descriptor set to all zeroes is an SGL Data Block descriptor with the Address field cleared to 0h and the Length field cleared to 0h may be used as a NULL descriptor.

Figure 115: SGL Descriptor Type

Code	Descriptor
0h	SGL Data Block descriptor
1h	SGL Bit Bucket descriptor
2h	SGL Segment descriptor
3h	SGL Last Segment descriptor

Figure 115: SGL Descriptor Type

Code	Descriptor
4h	Keyed SGL Data Block descriptor
5h	Transport SGL Data Block descriptor
6h to Eh	Reserved
Fh	Vendor specific

Figure 116 defines the SGL Descriptor Sub Type values and indicates the SGL Descriptor Types to which each SGL Descriptor Sub Type applies.

Figure 116: SGL Descriptor Sub Type Values

SGL Descriptor Sub Type	SGL Descriptor Types	Sub Type Description
0h	0h, 2h, 3h, 4h	**Address:** The Address field specifies the starting 64-bit memory byte address of the Data Block, Segment, or Last Segment descriptor.
	1h	For Type 1h, the Sub Type field shall be cleared to 0h.
	All other values	Reserved
1h	0h, 2h, 3h	**Offset:** The Address field contains an offset from the beginning of the location where data may be transferred. For NVMe over PCIe implementations, this Sub Type is reserved. For NVMe over Fabrics implementations, refer to section 3.3.2.1.3.1.
	1h	The controller shall abort the command with the status code of SGL Descriptor Type Invalid.
	4h	The controller shall abort the command with the status code of SGL Descriptor Type Invalid.
	All other values	Reserved
Ah to Fh	All	**NVMe Transport Specific:** The definitions for this range of Sub Types are defined by the binding section for the associated NVMe Transport.
All other values	All	Reserved

The SGL Data Block descriptor, defined in Figure 117, describes a data block.

Figure 117: SGL Data Block descriptor

Bytes	Description
07:00	**Address:** If the SGL Identifier Descriptor Sub Type field is cleared to 0h, then the Address field specifies the starting 64-bit memory byte address of the data block. If the SGL Identifier Descriptor Sub Type field is set to 1h, then the Address field contains an offset from the beginning of the location where data may be transferred. If the controller requires dword alignment and granularity as indicated in the SGL Support (SGLS) field of the Identify Controller data structure (refer to Figure 275), then the two least significant bits shall be cleared to 00b. If dword alignment and granularity is required, the controller may report an error of Invalid Field in Command if bits 1:0 are not cleared to 00b. If the controller does not report an error of Invalid Field in Command, then the controller shall operate as if bits 1:0 are cleared to 00b.
11:08	**Length:** The Length field specifies the length in bytes of the data block. A Length field cleared to 0h specifies that no data is transferred. An SGL Data Block descriptor specifying that no data is transferred is a valid SGL Data Block descriptor. If the controller requires dword alignment and granularity as specified in the SGL Support (SGLS) field the of Identify Controller data structure, then the two least significant bits shall be cleared to 00b. If dword alignment and granularity is required, the controller may report an error of Invalid Field in Command if bits 1:0 are not cleared to 00b. If the controller does not report an error of Invalid Field in Command, then the controller shall operate as if bits 1:0 are cleared to 00b. If the value in the Address field plus the value in the Length field is greater than 1_00000000_00000000h, then the SGL Data Block descriptor shall be processed as having a Data SGL Length Invalid or Metadata SGL Length Invalid error.

Figure 117: SGL Data Block descriptor

Bytes	Description
14:12	Reserved
15	**SGL Identifier:** The definition of this field is described in the table below. **Bits** / **Description** 03:00 / **SGL Descriptor Sub Type:** Valid values are specified in Figure 116. 07:04 / **SGL Descriptor Type:** 0h as specified in Figure 115.

The SGL Bit Bucket descriptor, defined in Figure 118, is used to ignore parts of source data.

Figure 118: SGL Bit Bucket descriptor

Bytes	Description
07:00	Reserved
11:08	**Length:** The Length field specifies the amount of source data that is discarded. An SGL Bit Bucket descriptor specifying that no source data be discarded (i.e., the length field cleared to 0h) is a valid SGL Bit Bucket descriptor. If the SGL Bit Bucket descriptor describes a destination data buffer (e.g., a read from the controller to host memory), then the Length field specifies the number of bytes of the source data which the controller shall discard (i.e., not transfer to the destination data buffer). If the SGL Bit Bucket descriptor describes a source data buffer (e.g., a write from host memory to the controller), then the Bit Bucket Descriptor shall be treated as if the Length field were cleared to 0h (i.e., the Bit Bucket Descriptor has no effect). If SGL Bit Bucket descriptors are supported, their length in a destination data buffer shall be included in the specified length of data to be transferred (e.g., their length in a source data buffer is not included in the transfer length specified by the NLB parameter).
14:12	Reserved
15	**SGL Identifier:** The definition of this field is described in the table below. **Bits** / **Description** 03:00 / **SGL Descriptor Sub Type:** Valid values are specified in Figure 116. 07:04 / **SGL Descriptor Type:** 1h as specified in Figure 115.

The SGL Segment descriptor, defined in Figure 119, describes the next SGL segment, which is not the last SGL segment.

Figure 119: SGL Segment descriptor

Bytes	Description
07:00	**Address:** If the SGL Identifier Descriptor Sub Type field is cleared to 0h, then the Address field specifies the starting 64-bit memory byte address of the next SGL segment, which is an SGL segment. If the SGL Identifier Descriptor Sub Type field is set to 1h, then the Address field contains an offset from the beginning of the location where data may be transferred.
11:08	**Length:** The Length field specifies the length in bytes of the next SGL segment. The Length field shall be a non-zero value and a multiple of 16. If the value in the Address field plus the value in the Length field is greater than 1_00000000_00000000h, then the SGL Segment descriptor shall be processed as having a Data SGL Length Invalid or Metadata SGL Length Invalid error.
14:12	Reserved
15	**SGL Identifier:** The definition of this field is described in the table below. **Bits** / **Description** 03:00 / **SGL Descriptor Sub Type:** Valid values are specified in Figure 116. 07:04 / **SGL Descriptor Type:** 2h as specified in Figure 115.

The SGL Last Segment descriptor, defined in Figure 120, describes the next and last SGL segment. A last SGL segment that contains an SGL Segment descriptor or an SGL Last Segment descriptor is processed as an error.

Figure 120: SGL Last Segment descriptor

Bytes	Description
07:00	**Address:** If the SGL Identifier Descriptor Sub Type field is cleared to 0h, then the Address field specifies the starting 64-bit memory byte address of the next and last SGL segment, which is an SGL segment. If the SGL Identifier Descriptor Sub Type field is set to 1h, then the Address field contains an offset from the beginning of the location where data may be transferred.
11:08	**Length:** The Length field specifies the length in bytes of the next and last SGL segment. The Length field shall be a non-zero value and a multiple of 16. If the value in the Address field plus the value in the Length field is greater than 1_00000000_00000000h, then the SGL Last Segment descriptor shall be processed as having a Data SGL Length Invalid or Metadata SGL Length Invalid error.
14:12	Reserved
15	**SGL Identifier:** The definition of this field is described in the table below. **Bits** / **Description** 03:00 — **SGL Descriptor Sub Type:** Valid values are specified in Figure 116. 07:04 — **SGL Descriptor Type:** 3h as specified in Figure 115.

The Keyed SGL Data Block descriptor, defined in Figure 121, describes a keyed data block.

Figure 121: Keyed SGL Data Block descriptor

Bytes	Description
07:00	**Address:** The Address field specifies the starting 64-bit memory byte address of the data block.
10:08	**Length:** The Length field specifies the length in bytes of the data block. A Length field cleared to 0h specifies that no data is transferred. An SGL Data Block descriptor specifying that no data is transferred is a valid SGL Data Block descriptor. If the value in the Address field plus the value in the Length field is greater than 1_00000000_00000000h, then the SGL Data Block descriptor shall be processed as having a Data SGL Length Invalid or Metadata SGL Length Invalid error.
14:11	**Key:** Specifies a 32-bit key that is associated with the data block.
15	**SGL Identifier:** The definition of this field is described in the table below. **Bits** / **Description** 03:00 — **SGL Descriptor Sub Type:** Valid values are specified in Figure 116. 07:04 — **SGL Descriptor Type:** 4h as specified in Figure 115.

Figure 122: Transport SGL Data Block descriptor

Bytes	Description
07:00	Reserved
11:08	**Length:** The Length field specifies the length in bytes of the data block. A Length field cleared to 0h specifies that no data is transferred. A Transport SGL Data Block descriptor specifying that no data is transferred is a valid Transport SGL Data Block descriptor. If the controller requires dword alignment and granularity as specified in the SGL Support field of Identify Controller (refer to Figure 275), then the two least significant bits shall be cleared to 00b. The data transfer mechanism and data buffers for data specified by a Transport SGL Data Block descriptor are defined by the binding section for the associated NVMe Transport.
14:12	Reserved

Figure 122: Transport SGL Data Block descriptor

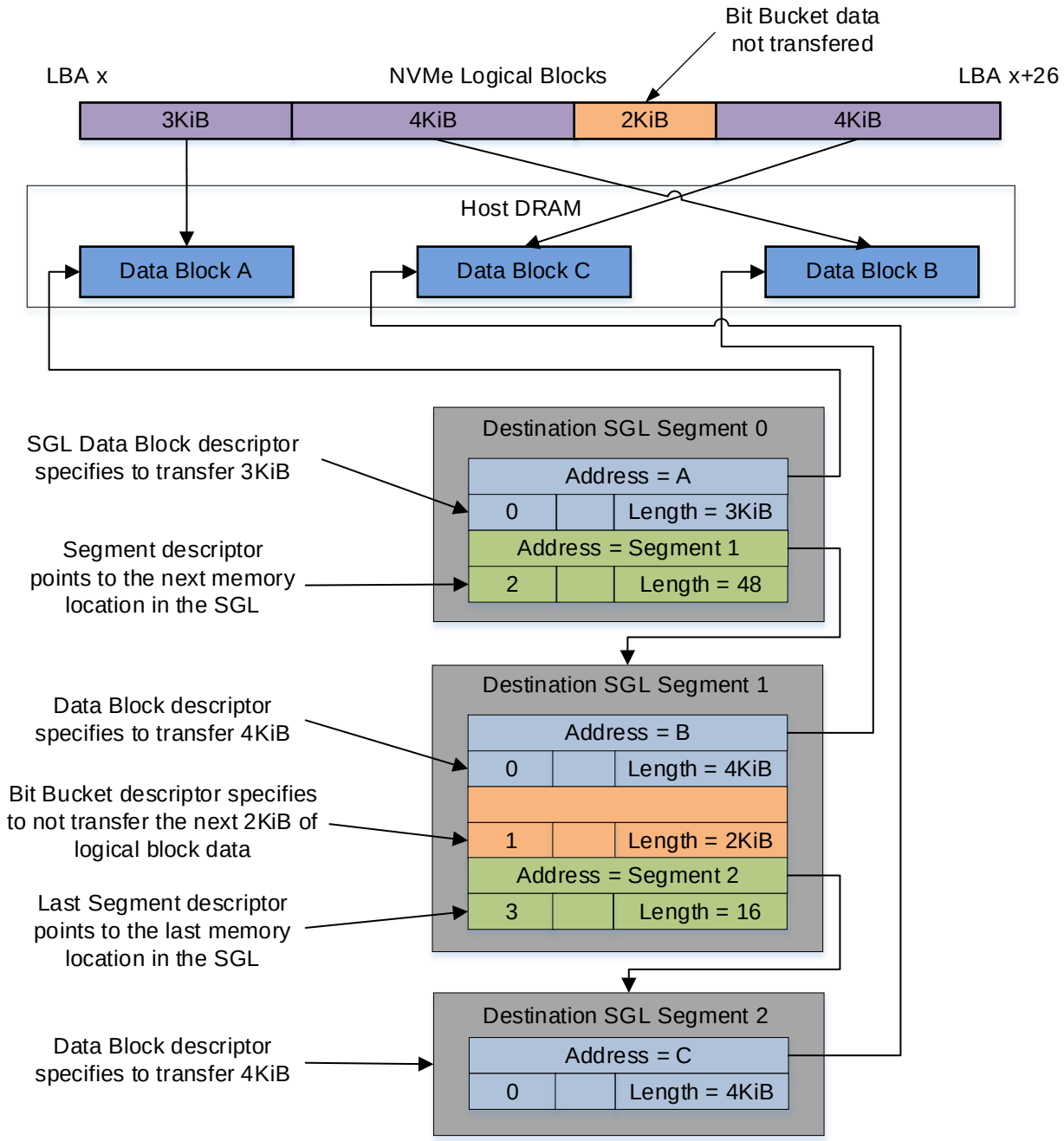
Bytes	Description
15	**SGL Identifier:** The definition of this field is described in the table below. **Bits** / **Description** 03:00 – **SGL Descriptor Sub Type:** Valid values are specified in Figure 116. 07:04 – **SGL Descriptor Type:** 5h as specified in Figure 115.

4.1.2.1 SGL Example

Figure 123 shows an example of a data read request using SGLs. In the example, the logical block size is 512B. The total length of the logical blocks accessed is 13 KiB, of which only 11 KiB is transferred to the host. The Number of Logical Blocks (NLB) field in the command shall specify 26, indicating the total length of the logical blocks accessed on the controller is 13 KiB. There are three SGL segments describing the locations in memory where the logical block data is transferred.

The three SGL segments contain a total of three Data Block descriptors with lengths of 3 KiB, 4 KiB, and 4 KiB respectively. Segment 1 of the Destination SGL contains a Bit Bucket descriptor with a length of 2 KiB that specifies to not transfer (i.e., ignore) 2 KiB of logical block data from the NVM. Segment 1 of the destination SGL also contains a Last Segment descriptor specifying that the segment pointed to by the descriptor is the last SGL segment.

Figure 123: SGL Read Example



4.1.3 Metadata Region (MR)

The definition for the Metadata Region is command set specific. Refer to each I/O Command Set specification for applicability and additional details, if any.

4.2 Feature Values

The Get Features command (refer to section 5.15) and Set Features command (refer to section 5.27) may be used to read and modify operating parameters of the controller. The operating parameters are grouped and identified by Feature Identifiers. Each Feature Identifier contains one or more attributes that may affect the behavior of the Feature.

If bit 4 is set to ‘1’ in the Optional NVM Command Support (ONCS) field of the Identify Controller data structure in Figure 275, then for each Feature, there are three settings: default, saved, and current. If bit 4 is cleared to ‘0’ in the Optional NVM Command Support field of the Identify Controller data structure, then the controller only supports a current and default value for each Feature. In this case, the current value may be persistent across power cycles and resets based on the information specified in Figure 316.

If bit 4 is set to ‘1’ in the ONCS field, then each Feature has supported capabilities (refer to Figure 195), which are discovered using the Supported Capabilities value in the Select field in Get Features (refer to Figure 192).

The default value for each Feature is vendor specific and set by the manufacturer unless otherwise specified. The default value is not changeable.

A Feature may be saveable. The saved value is the value that the Feature has after a Controller Level Reset. If a Feature is not saveable, then:

a) the default value is used after a Controller Level Reset; and
b) a Get Features command to read the saved value returns the default value.

The current value for a Feature is the value in active use by the controller for that Feature.

A Set Features command uses the value specified by the command to set:

a) the current value for that Feature; or
b) the current value for that Feature and the saved value for that Feature, if that Feature is saveable.

Feature settings may apply to:

a) the controller (i.e., the feature is not namespace specific); or
b) a namespace (i.e., the feature is namespace specific).

For feature values that apply to the controller:

a) if the NSID field is cleared to 0h or set to FFFFFFFFh, then:

   - the Set Features command shall set the specified feature value for the controller; and
   - the Get Features command shall return the current setting of the requested feature value for the controller;

   and

b) if the NSID field is set to a valid namespace identifier (refer to section 3.2.1.2), then:

   - the Set Features command shall abort with a status code of Feature Not Namespace Specific; and
   - the Get Features command shall return the current setting of the requested feature value for the controller.

For feature values that apply to a namespace:

a) if the NSID field is set to an active namespace identifier (refer to section 3.2.1.4), then:

   - the Set Features command shall set the specified feature value of the specified namespace; and
   - the Get Features command shall return the current setting of the requested feature value for the specified namespace;

b) if the NSID field is set to FFFFFFFFh, then:

   - for the Set Features command, the controller shall:
     - if the MDS bit is set to ‘1’ in the Identify Controller data structure, abort the command with Invalid Field in Command; or

- if the MDS bit is cleared to '0' in the Identify Controller data structure, unless otherwise specified, set the specified feature value for all namespaces attached to the controller processing the command;

and

- for the Get Features command, the controller shall, unless otherwise specified in section 5.27.1, abort the command with a status code of Invalid Namespace or Format;

and

c) if the NSID field is set to any other value, then the Set Features command and the Get Features command shall abort as described in the rules for namespace identifier usage in Figure 87.

If the controller supports the Save field in the Set Features command and the Select field in the Get Features command, then any Feature Identifier that is namespace specific may be saved on a per namespace basis.

There are mandatory and optional Feature Identifiers defined in Figure 316. If a Get Features command or Set Features command is processed that specifies a Feature Identifier that is not supported, then the controller shall abort the command with a status code of Invalid Field in Command.

4.3 Identifier Format and Layout (Informative)

This section provides guidance for proper implementation of various identifiers defined in the Identify Controller, Identify Namespace, and Namespace Identification Descriptor data structures.

4.3.1 PCI Vendor ID (VID) and PCI Subsystem Vendor ID (SSVID)

The PCI Vendor ID (VID, bytes 01:00) and PCI Subsystem Vendor ID (SSVID, bytes 03:02) are defined in the Identify Controller data structure. The values are assigned by the PCI SIG. Each identifier is a 16-bit number in little endian format.

Example:

- VID = ABCDh; and
- SSVID = 1234h.

Figure 124: PCI Vendor ID (VID) and PCI Subsystem Vendor ID (SSVID)

Byte	00	01	02	03
Value	CDh	ABh	34h	12h

4.3.2 Serial Number (SN) and Model Number (MN)

The Serial Number (SN, bytes 23:04) and Model Number (MN, bytes 63:24) are defined in the Identify Controller data structure. The values are ASCII strings assigned by the vendor. Each identifier is in big endian format.

Example (Value shown as ASCII characters):

- SN = "SN1"; and
- MN = "M2".

Figure 125: Serial Number (SN) and Model Number (MN)

Bytes	04	05	06	23 to 07	24	25	63 to 26
Value	53h ('S')	4Eh ('N')	31h ('1')	20h (' ')	4Dh ('M')	32h ('2')	20h (' ')

4.3.3 IEEE OUI Identifier (IEEE)

The IEEE OUI Identifier (OUI, bytes 75:73) is defined in the Identify Controller data structure. The value is assigned by the IEEE Registration Authority. The identifier is in little endian format.

Example:

- OUI = ABCDEFh.

Figure 126: IEEE OUI Identifier (IEEE)

Byte	73	74	75
Value	EFh	CDh	ABh

4.3.4 IEEE Extended Unique Identifier (EUI64)

The IEEE Extended Unique Identifier (EUI64, bytes 127:120) is defined in the Identify Namespace data structure. Tutorials are available at https://standards.ieee.org/develop/regauth/tut/index.html. IEEE defines three formats that may be used in this field: MA-L, MA-M, and MA-S. The examples in this section use the MA-L format.

The MA-L format is defined as a string of eight octets:

Figure 127: IEEE Extended Unique Identifier (EUI64), MA-L Format

EUI[0]	EUI[1]	EUI[2]	EUI[3]	EUI[4]	EUI[5]	EUI[6]	EUI[7]
OUI			Extension Identifier				

EUI64 is defined in big endian format. The OUI field differs from the OUI Identifier which is in little endian format as described in section 4.3.3.

Example:

- OUI Identifier = ABCDEFh; and
- Extension Identifier = 0123456789h.

Figure 128: IEEE Extended Unique Identifier (EUI64), OUI Identifier

Byte	120	121	122	123	124	125
Value	ABh	CDh	EFh	01h	23h	45h
Field	OUI			Extension Identifier		

Figure 129: IEEE Extended Unique Identifier (EUI64), Ext. ID (cont)

Byte	126	127
Value	67h	89h
Field	Ext ID (cont)	

The MA-L format is similar to the World Wide Name (WWN) format defined as IEEE Registered designator (NAA = 5) as shown below.

Figure 130: MA-L similarity to WWN

Byte	0	1	2	3	4	5	6	7
EUI64	OUI			Extension Identifier				
WWN (NAA = 5)	5h	OUI		Vendor Specific Identifier				

4.3.5 Namespace Globally Unique Identifier (NGUID)

The Namespace Globally Unique Identifier (NGUID, bytes 119:104) is defined in the Identify Namespace data structure. The NGUID is composed of an IEEE OUI, an extension identifier, and a vendor specific extension identifier. The extension identifier and vendor specific extension identifier are both assigned by

the vendor and may be considered as a single field. NGUID is defined in big endian format. The OUI field differs from the OUI Identifier which is in little endian format as described in section 4.3.3.

Example:

- OUI Identifier = ABCDEFh;
- Extension Identifier = 0123456789h; and
- Vendor Specific Extension Identifier = FEDCBA9876543210h.

Figure 131: Namespace Globally Unique Identifier (NGUID)

Byte	104	105	106	107	108	109
Value	FEh	DCh	BAh	98h	76h	54h
Field	Vendor Specific Extension Identifier					

Figure 132: Namespace Globally Unique Identifier (NGUID), OUI

Byte	110	111	112	113	114	115
Value	32h	10h	ABh	CDh	EFh	01h
Field	VSP Ex ID (cont)		OUI			Ex ID

Figure 133: Namespace Globally Unique Identifier (NGUID), Extension Identifier (continued)

Byte	116	117	118	119
Value	23h	45h	67h	89h
Field	Extension Identifier (continued)			

The NGUID format is similar to the World Wide Name (WWN) format as IEEE Registered Extended designator (NAA = 6) as shown below.

Figure 134: Namespace Globally Unique Identifier (NGUID), NGUID similarity to WWN

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NGUID	Vendor Specific Extension Identifier								OUI			Extension Identifier				
WWN (NAA = 6)	6h	OUI			Vendor Specific Identifier				Vendor Specific Identifier Extension							

4.3.6 Universally Unique Identifier (UUID)

The Universally Unique Identifier is defined in RFC 4122 and contained in the Namespace Identification Descriptor (refer to Figure 277). Byte ordering requirements for a UUID are described in RFC 4122.

4.4 List Data Structures

This section describes list data structures used in this specification.

4.4.1 Controller List

A Controller List, defined in Figure 135, is an ordered list of ascending controller IDs. The controller identifier is defined in bytes 79:78 of the Identify data structure in Figure 135. Unused entries are zero filled.

Figure 135: Controller List Format

Bytes	Description
01:00	**Number of Identifiers:** This field contains the number of controller entries in the list. There may be up to 2,047 identifiers in the list. A value of 0h indicates there are no controllers in the list.
03:02	**Identifier 0:** This field contains the NVM subsystem unique controller identifier for the first controller in the list, if present.

Figure 135: Controller List Format

Bytes	Description
05:04	**Identifier 1:** This field contains the NVM subsystem unique controller identifier for the second controller in the list, if present.
…	…
(N*2+3):(N*2+2)	**Identifier N:** This field contains the NVM subsystem unique controller identifier for the N+1 controller in the list, if present.

4.4.2 Namespace List

A Namespace List, defined in Figure 136, is an ordered list of namespace IDs. Unused entries are zero filled.

Figure 136: Namespace List Format

Bytes	Description
03:00	**Identifier 0:** This field contains the lowest namespace ID in the list or 0h if the list is empty.
07:04	**Identifier 1:** This field contains the second lowest namespace ID in the list or 0h if the list contains less than two entries.
…	…
(N*4+3):(N*4)	**Identifier N:** This field contains the N+1 lowest namespace ID in the list or 0h if the list contains fewer than N entries.

4.5 NVMe Qualified Names

NVMe Qualified Names (NQNs) are used to uniquely describe a host or NVM subsystem for the purposes of identification and authentication. The NVMe Qualified Name for the NVM subsystem is specified in the Identify Controller data structure. An NQN is permanent for the lifetime of the host or NVM subsystem.

An NVMe Qualified Name is encoded as a string of Unicode characters with the following properties:

- The encoding is UTF-8 (refer to RFC 3629);
- The following characters are used in formatting:
  - dash (‘-‘=U+002d);
  - dot (‘.’=U+002e); and
  - colon (‘:’=U+003a);
- The maximum name is 223 bytes in length; and
- The string is null terminated.

There are two supported NQN formats. The first format may be used by any organization that owns a domain name. This naming format may be used to create a human interpretable string to describe the host or NVM subsystem. This format consists of:

- The string “nqn”;
- The string “.” (i.e., the ASCII period character);
- A date code, in “yyyy-mm” format. This date shall be during a time interval when the naming authority owned the domain name used in this format. The date code uses the Gregorian calendar. All digits and the dash shall be included;
- The string “.” (i.e., the ASCII period character);
- The reverse domain name of the naming authority that is creating the NQN; and
- A colon (:) prefixed string that the owner of the domain name assigns that does not exceed the maximum length. The naming authority is responsible to ensure that the NQN is worldwide unique.

The reverse domain name in an NQN that uses this format shall not be “org.nvmexpress”.

The following are examples of NVMe Qualified Names that may be generated by “Example NVMe, Inc.”

- The string “nqn.2014-08.com.example:nvme:nvm-subsystem-sn-d78432”; and
- The string “nqn.2014-08.com.example:nvme.host.sys.xyz”.

The second format may be used to create a unique identifier when there is not a naming authority or there is not a requirement for a human interpretable string. This format consists of:

- The string “nqn”;
- The string “.” (i.e., the ASCII period character);
- The string “2014-08.org.nvmexpress:uuid:”; and
- A 128-bit UUID based on the definition in RFC 4122 represented as a string formatted as “*11111111-2222-3333-4444-555555555555*”.

The following is an example of an NVMe Qualified Name using the UUID-based format:

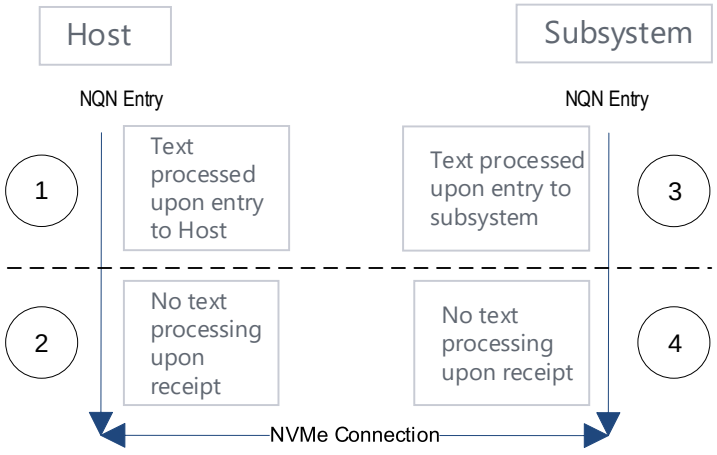
- The string “nqn.2014-08.org.nvmexpress:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6”.

NVMe hosts, controllers and NVM subsystems compare (e.g., for equality) NVMe Qualified Names used by NVMe as binary strings without any text processing or text comparison logic that is specific to the Unicode character set or locale (e.g., case folding or conversion to lower case, Unicode normalization). Any such text processing:

a) may occur as part of entry of NVMe Qualified Names into NVMe hosts and NVM subsystems; and
b) should not occur as part of receiving NVMe Qualified Names via an NVMe connection, as shown in Figure 137.

Upon entry (e.g., at point 1 in Figure 137, described as “input” in RFC4122), NVMe host software may process an NVMe Qualified Name (e.g., for conversion to lower case based on the Unicode locale). Upon entry (e.g., at point 3 in Figure 137, described as “input” in RFC4122), a controller may process an NVMe Qualified Name (e.g., for conversion to lower case based on the Unicode locale). Upon receipt by the host (e.g., at point 2 in Figure 137) of an NVMe Qualified Name from the controller, no text process (e.g., no case folding) should occur. Upon receipt by the controller (e.g., at point 4 in Figure 137) of an NVMe Qualified Name from the host, no text processing (e.g., no case folding) should occur.

Figure 137: NQN Processing



4.5.1 Unique Identifier

The NVM Subsystem NVMe Qualified Name specified in the Identify Controller data structure (refer to Figure 275) should be used (e.g., by host software) as the unique identifier for the NVM subsystem. If the controller complies with an older version of the NVM Express specification that does not include the NVM Subsystem NQN, then the PCI Vendor ID, Serial Number, and Model Number fields in the Identify Controller data structure and the NQN Starting String “nqn.2014.08.org.nvmexpress:” may be combined by the host

to form a globally unique value that identifies the NVM subsystem (e.g., for host software that uses NQNs). The method shown in Figure 138 should be used by the host to construct an NVM Subsystem NQN for older NVM subsystems that do not provide an NQN in the Identify Controller data structure. The mechanism used by the vendor to assign Serial Number and Model Number values to ensure uniqueness is outside the scope of this specification.

Figure 138: NQN Construction for Older NVM Subsystems

Bytes	Description
26:00	**NQN Starting String (NSS):** Contains the 27 letter ASCII string ”nqn.2014-08.org.nvmexpress:”.
30:27	**PCI Vendor ID (VID):** Contains the company vendor identifier that is assigned by the PCI SIG as a hexadecimal ASCII string.
34:31	**PCI Subsystem Vendor ID (SSVID):** Contains the company vendor identifier that is assigned by the PCI SIG for the subsystem as a hexadecimal ASCII string.
54:35	**Serial Number (SN):** Contains the serial number for the NVM subsystem that is assigned by the vendor as an ASCII string.
94:55	**Model Number (MN):** Contains the model number for the NVM subsystem that is assigned by the vendor as an ASCII string.
222:95	**Padding (PAD):** Contains spaces (ASCII character 20h).

An NVM subsystem may contain multiple controllers. All controllers contained in the NVM subsystem share the same NVM subsystem unique identifier. The Controller ID (CNTLID) value returned in the Identify Controller data structure may be used to uniquely identify a controller within an NVM subsystem. The Controller ID value when combined with the NVM subsystem identifier forms a globally unique value that identifies the controller. The mechanism used by the vendor to assign Controller ID values is outside the scope of this specification.

The Identify Namespace data structure (refer to the applicable I/O Command Set specification) contains the IEEE Extended Unique Identifier (EUI64) and the Namespace Globally Unique Identifier (NGUID) fields. The Namespace Identification Descriptor data structure (refer to Figure 277) contains the Namespace UUID. EUI64 is an 8-byte EUI-64 identifier (refer to section 4.3.4), NGUID is a 16-byte identifier based on EUI-64 (refer to section 4.3.5), and Namespace UUID is a 16-byte identifier described in RFC 4122 (refer to section 4.3.6).

When creating a namespace, the controller shall indicate a globally unique value in one or more of the following:

a) the EUI64 field;
b) the NGUID field; or
c) a Namespace Identification Descriptor with the Namespace Identifier Type field set to 3h.

If the EUI64 field is cleared to 0h and the NGUID field is cleared to 0h, then the namespace shall support a valid Namespace UUID in the Namespace Identification Descriptor data structure.
If the UIDREUSE bit in the NSFEAT field is cleared to ‘0’, then a controller may reuse a non-zero NGUID/EUI64 value for a new namespace after the original namespace using the value has been deleted. If the UIDREUSE bit is set to ‘1’, then a controller shall not reuse a non-zero NGUID/EUI64 for a new namespace after the original namespace using the value has been deleted.

5 Admin Command Set

The Admin Command Set defines the commands that may be submitted to the Admin Submission Queue.

The submission queue entry (SQE) structure and the fields that are common to all Admin commands are defined in section 3.3.3. The completion queue entry (CQE) structure and the fields that are common to all Admin commands are defined in section 3.3.3.2. The command specific fields in the SQE and CQE structures (i.e., SQE Command Dwords 10 to 15, CQE Dword 0, and CQE Dword 1) for the Admin Command Set are defined in this section.

Admin commands should not be impacted by the state of I/O queues (e.g., a full I/O Completion Queue should not delay or stall the Delete I/O Submission Queue command).

Figure 139 defines all Admin commands. Refer to Figure 22, Figure 28, and Figure 32 for mandatory, optional, and prohibited commands for the various controller types.

Figure 139: Opcodes for Admin Commands

Opcode by Field			Combined Opcode [1]	Namespace Identifier Used [2]	Command	Command Set Specific [8]
(07)	(06:02)	(01:00)				
Generic Command	Function	Data Transfer[3]				
0b	000 00b	00b	00h	No	Delete I/O Submission Queue	No
0b	000 00b	01b	01h	No	Create I/O Submission Queue	No
0b	000 00b	10b	02h	Yes	Get Log Page	No
0b	000 01b	00b	04h	No	Delete I/O Completion Queue	No
0b	000 01b	01b	05h	No	Create I/O Completion Queue	No
0b	000 01b	10b	06h	NOTE 6	Identify	No
0b	000 10b	00b	08h	No	Abort	No
0b	000 10b	01b	09h	Yes	Set Features	No
0b	000 10b	10b	0Ah	Yes	Get Features	No
0b	000 11b	00b	0Ch	No	Asynchronous Event Request	No
0b	000 11b	01b	0Dh	Yes	Namespace Management	No
0b	001 00b	00b	10h	No	Firmware Commit	No
0b	001 00b	01b	11h	No	Firmware Image Download	No
0b	001 01b	00b	14h	Yes	Device Self-test	No
0b	001 01b	01b	15h	Yes [4]	Namespace Attachment	No
0b	001 10b	00b	18h	No	Keep Alive	No
0b	001 10b	01b	19h	Yes [5]	Directive Send	No
0b	001 10b	10b	1Ah	Yes [5]	Directive Receive	No
0b	001 11b	00b	1Ch	No	Virtualization Management	No
0b	001 11b	01b	1Dh	No	NVMe-MI Send	No
0b	001 11b	10b	1Eh	No	NVMe-MI Receive	No
0b	010 00b	00b	20h	No	Capacity Management	No
0b	010 01b	00b	24h	No	Lockdown	No
0b	111 11b	00b	7Ch	No	Doorbell Buffer Config	No
0b	111 11b	11b	7Fh	NOTE 9	Fabrics Commands[9]	No
1b	000 00b	00b	80h	Yes	Format NVM	No
1b	000 00b	01b	81h	NOTE 7	Security Send	No
1b	000 00b	10b	82h	NOTE 7	Security Receive	No
1b	000 01b	00b	84h	No	Sanitize	No
1b	000 01b	10b	86h	NOTE 4	Get LBA Status	NVM, ZNS

Figure 139: Opcodes for Admin Commands

Opcode by Field			Combined Opcode [1]	Namespace Identifier Used [2]	Command	Command Set Specific [8]
(07)	(06:02)	(01:00)				
Generic Command	Function	Data Transfer[3]				
	Vendor Specific					
1b	n/a	NOTE 3	C0h to FFh		Vendor specific	

Notes:
1. Opcodes not listed are reserved.
2. A subset of commands use the Namespace Identifier (NSID) field. If the Namespace Identifier field is used, then the value FFFFFFFFh is supported in this field unless otherwise indicated in footnotes in this figure that a specific command does not support that value or supports that value only under specific conditions. When this field is not used, the field is cleared to 0h as described in Figure 87.
3. Indicates the data transfer direction of the command. All options to the command shall transfer data as specified or transfer no data. All commands, including vendor specific commands, shall follow this convention: 00b = no data transfer; 01b = host to controller; 10b = controller to host; 11b = bidirectional.
4. This command does not support the use of the Namespace Identifier (NSID) field set to FFFFFFFFh.
5. Support for the Namespace Identifier field set to FFFFFFFFh depends on the Directive Operation (refer to section 8.7).
6. Use of the Namespace Identifier field depends on the CNS value in the Identify Command as described in Figure 273.
7. The use of the Namespace Identifier is Security Protocol specific.
8. No = Not I/O Command Set specific, A = All I/O Command Sets, NVM = NVM Command Set specific, ZNS = Zoned Namespace Command Set.
9. All Fabrics commands use the opcode 7Fh. Refer to section 6 for details.

Figure 140 lists the Admin commands that are allowed during the processing of a sanitize operation and the Admin commands that should be allowed during the processing of a Format NVM command .

If a Format NVM command is in progress, then an Admin command not listed in Figure 140 that is submitted for any namespace affected by that Format NVM command may be aborted. If aborted for that reason, then a status code of Format in Progress should be returned.

If there are Admin commands not listed in Figure 140 being processed for a namespace, then a Format NVM command which is submitted that affects that namespace may be aborted. If aborted for that reason, then a status code of Command Sequence Error should be returned.

Figure 140: Sanitize Operations and Format NVM Command – Admin Commands Allowed

Admin Command	Additional Restrictions for Format NVM command	Additional Restrictions for sanitize operations
Abort		
Asynchronous Event Request		
Create I/O Completion Queue		
Create I/O Submission Queue		
Device Self–test	Only Controller DST should be allowed	Prohibited
Delete I/O Completion Queue		
Delete I/O Submission Queue		
Get Features		

Figure 140: Sanitize Operations and Format NVM Command – Admin Commands Allowed

<table>
<tr><th>Admin Command</th><th>Additional Restrictions for Format NVM command</th><th>Additional Restrictions for sanitize operations</th></tr>
<tr><td>Get Log Page</td><td colspan="2">The log pages allowed for both Format NVM command and sanitize operations are listed below.
<table>
<tr><th>Log Pages</th><th>Additional Restrictions for both Format NVM command and sanitize operations</th></tr>
<tr><td>Error Information</td><td>Return 0h in the LBA field.</td></tr>
<tr><td>SMART / Health Information</td><td></td></tr>
<tr><td>Changed Namespace List</td><td></td></tr>
<tr><td>Reservation Notification</td><td></td></tr>
<tr><td>Asymmetric Namespace Access</td><td></td></tr>
<tr><td>Sanitize Status</td><td></td></tr>
<tr><td>Vendor specific</td><td>Prohibited for Sanitize</td></tr>
<tr><td>Persistent Event Log</td><td>Prohibited for Sanitize</td></tr>
<tr><td>Boot Partition</td><td></td></tr>
</table></td></tr>
<tr><td>Identify</td><td></td><td></td></tr>
<tr><td>Keep Alive</td><td></td><td></td></tr>
<tr><td>NVMe-MI Receive</td><td colspan="2" rowspan="2">Allowed unless explicitly prohibited in the NVM Express Management Interface Specification.</td></tr>
<tr><td>NVMe-MI Send</td></tr>
<tr><td>Sanitize</td><td></td><td>Prohibited</td></tr>
<tr><td>Set Features</td><td colspan="2">Namespace Write Protection Config Feature is not allowed.</td></tr>
<tr><td>Opcode 7Fh</td><td colspan="2">The Fabric Commands allowed are listed below. Refer to section 6.
<table>
<tr><th>Fabrics Commands</th><th>Additional Restrictions for both Format NVM command and sanitize operations</th></tr>
<tr><td>Property Set</td><td></td></tr>
<tr><td>Connect</td><td></td></tr>
<tr><td>Disconnect</td><td></td></tr>
<tr><td>Property Get</td><td></td></tr>
<tr><td>Authentication Send</td><td></td></tr>
<tr><td>Authentication Receive</td><td></td></tr>
<tr><td>Vendor Specific</td><td>Commands are allowed that do not affect or retrieve user data.</td></tr>
</table></td></tr>
<tr><td>Vendor Specific</td><td colspan="2">Commands are allowed that do not affect or retrieve user data.</td></tr>
</table>

5.1 Abort command

The Abort command is used to abort a specific command previously submitted to the Admin Submission Queue or an I/O Submission Queue. An Abort command is a best effort command; the command to abort may have already completed, currently be in execution, or may be deeply queued.

To abort a large number of commands (e.g., a larger number of commands than the limit listed in the ACL field), the host should follow the procedures described in section 3.7.3 to delete the I/O Submission Queue and recreate the I/O Submission Queue.

The Abort command uses the Command Dword 10 field. All other command specific fields are reserved.

The Abort Command Limit field in the Identify Controller data structure (refer to Figure 275) indicates the controller limit on concurrent execution of Abort commands. A host should not allow the number of outstanding Abort commands to exceed this value. The controller may complete any excess Abort commands with Abort Command Limit Exceeded status.

Figure 141: Abort – Command Dword 10

Bits	Description
31:16	**Command Identifier (CID):** This field specifies the command identifier of the command to be aborted, that was specified in the CDW0.CID field within the command itself.
15:00	**Submission Queue Identifier (SQID):** This field specifies the identifier of the Submission Queue that the command to be aborted is associated with.

5.1.1 Command Completion

Upon completion of the Abort command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the Abort command and indicating whether the command to abort was aborted. Dword 0 of the completion queue entry indicates whether the command to abort was aborted.

If the command to abort was successfully aborted, then a completion queue entry for the aborted command shall be posted to the appropriate Admin or I/O Completion Queue with a status code of Command Abort Requested before the completion queue entry for the Abort command is posted to the Admin Completion Queue, and bit 0 of Dword 0 shall be cleared to '0' in the completion queue entry for the Abort command. If the command to abort was not aborted for any reason, then bit 0 of Dword 0 shall be set to '1' in the completion queue entry for the Abort command.

Command specific status values associated with the Abort command are defined in Figure 142.

Figure 142: Abort – Command Specific Status Values

Value	Description
3h	**Abort Command Limit Exceeded:** The number of concurrently outstanding Abort commands has exceeded the limit indicated in the Identify Controller data structure.

5.2 Asynchronous Event Request command

Asynchronous events are used to notify host software of status, error, and health information as these events occur. To enable asynchronous events to be reported by the controller, host software needs to submit one or more Asynchronous Event Request commands to the controller. The controller specifies an event to the host by completing an Asynchronous Event Request command. Host software should expect that the controller may not execute the command immediately; the command should be completed when there is an event to be reported.

The Asynchronous Event Request command is submitted by host software to enable the reporting of asynchronous events from the controller. This command has no timeout. The controller posts a completion queue entry for this command when there is an asynchronous event to report to the host. If Asynchronous Event Request commands are outstanding when the controller is reset, then each of those commands is aborted and should not return a CQE.

All command specific fields are reserved.

Host software may submit multiple Asynchronous Event Request commands to reduce event reporting latency. The total number of simultaneously outstanding Asynchronous Event Request commands is limited by the value indicated in the Asynchronous Event Request Limit field in the Identify Controller data structure in Figure 275.

Asynchronous events are grouped into event types. The event type is indicated in the Asynchronous Event Type field in Dword 0 of the completion queue entry for the Asynchronous Event Request command. When the controller posts a completion queue entry for an outstanding Asynchronous Event Request command and thus reports an asynchronous event, subsequent events of that event type are automatically masked by the controller until the host clears that event. Unless otherwise stated, an event is cleared by reading the log page associated with that event using the Get Log Page command (refer to section 5.16). If that log page is not accessible because media is not ready (i.e., the controller aborts the Get Log Page command with a status code of Admin Command Media Not Ready), then the controller shall not post a completion

queue entry for that asynchronous event until the controller is able to successfully return the log page that is required to be read to clear the asynchronous event.

The following event types are defined:

a) **Error event:** Indicates a general error that is not associated with a specific command (refer to Figure 145). To clear this event, host software reads the Error Information log page (refer to section 5.16.1.2) using the Get Log Page command with the Retain Asynchronous Event bit cleared to '0';
b) **SMART / Health Status event:** Indicates a SMART or health status event (refer to Figure 146). To clear this event, host software reads the SMART / Health Information log (refer to section 5.16.1.3) using the Get Log Page command with the Retain Asynchronous Event bit cleared to '0'. The SMART / Health conditions that trigger asynchronous events may be configured in the Asynchronous Event Configuration feature using the Set Features command (refer to section 5.27.1.8);
c) **Notice event:** Indicates a general event (refer to Figure 147). To clear this event, host software reads the appropriate log page as described in Figure 147. The conditions that trigger asynchronous events may be configured in the Asynchronous Event Configuration feature using the Set Features command (refer to section 5.27.1.8);
d) **I/O Command Specific Status events:** Events that are specific to an I/O command (refer to Figure 148);
e) **Immediate events:** Events that are only reported when an outstanding Asynchronous Event Request command exists at the time the event occurs. If the event occurs and there is no outstanding Asynchronous Event Request command, then the event shall not be reported. No log page is associated with these events. These events include:
    A. Normal NVM Subsystem Shutdown event;

    and

f) **Vendor Specific event:** Indicates a vendor specific event. To clear this event, host software reads the indicated vendor specific log page using the Get Log Page command with the Retain Asynchronous Event bit cleared to '0'.

The Sanitize Operation Completed With Unexpected Deallocation asynchronous event shall be supported if the controller supports the Sanitize Config feature (refer to section 5.27.1.19).

Asynchronous events are reported due to a new entry being added to a log page (e.g., Error Information log page) or a status update (e.g., status in the SMART / Health log page). A status change may be permanent (e.g., the media has become read only) or transient (e.g., the temperature reached or exceeded a threshold for a period of time). Host software should modify the event threshold or mask the event for transient and permanent status changes before issuing another Asynchronous Event Request command to avoid repeated reporting of asynchronous events.

If an event occurs for which reporting is enabled and there are no Asynchronous Event Request commands outstanding, the controller should retain the event information for that Asynchronous Event Type and use that information as a response to the next Asynchronous Event Request command that is received. If a Get Log Page command clears the event prior to receiving the Asynchronous Event Request command or if a power off condition occurs, then a notification is not sent. If multiple events of the same type occur that have identical responses to the Asynchronous Event Request command, then those events may be reported as a single response to an Asynchronous Event Request command. If multiple events occur that are of different types or have different responses to the Asynchronous Event Request command, then the controller should retain a queue of those events for reporting in responses to subsequent Asynchronous Event Request commands.

5.2.1 Command Completion

A completion queue entry is posted to the Admin Completion Queue if there is an asynchronous event to report to the host. Command specific status values associated with Asynchronous Event Request are defined in Figure 143.

Figure 143: Status Code – Command Specific Status Values

Value	Description
05h	**Asynchronous Event Request Limit Exceeded:** The number of concurrently outstanding Asynchronous Event Request commands has been exceeded.

Dword 0 of the completion queue entry contains information about the asynchronous event. The definition of Dword 0 of the completion queue entry is in Figure 144.

Figure 144: Asynchronous Event Request – Completion Queue Entry Dword 0

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:24</td><td>Reserved</td></tr>
<tr><td>23:16</td><td>Log Page Identifier: Indicates the log page associated with the asynchronous event. This log page needs to be read by the host to clear the event.</td></tr>
<tr><td>15:08</td><td>Asynchronous Event Information: Refer to Figure 145, Figure 146, Figure 147, and Figure 148 for detailed information regarding the asynchronous event.</td></tr>
<tr><td>07:03</td><td>Reserved</td></tr>
<tr><td>02:00</td><td>Asynchronous Event Type: Indicates the event type of the asynchronous event. More specific information on the event is provided in the Asynchronous Event Information field.
<table>
<tr><th>Value</th><th>Definition</th><th>Reference</th></tr>
<tr><td>000b</td><td>Error status</td><td>Figure 145</td></tr>
<tr><td>001b</td><td>SMART / Health status</td><td>Figure 146</td></tr>
<tr><td>010b</td><td>Notice</td><td>Figure 147</td></tr>
<tr><td>011b</td><td>Immediate</td><td>Figure 149</td></tr>
<tr><td>100b to 101b</td><td>Reserved</td><td>-</td></tr>
<tr><td>110b</td><td>I/O Command specific status</td><td>Figure 148</td></tr>
<tr><td>111b</td><td>Vendor specific</td><td>-</td></tr>
</table></td></tr>
</table>

The information in either Figure 145, Figure 146, Figure 147, or Figure 148 is returned in the Asynchronous Event Information field, depending on the Asynchronous Event Type.

Figure 145: Asynchronous Event Information – Error Status

Value	Description
00h	**Write to Invalid Doorbell Register:** Host software wrote the doorbell of a queue that was not created.
01h	**Invalid Doorbell Write Value:** Host software attempted to write an invalid doorbell value. Some possible causes of this error are: • the value written was out of range of the corresponding queue's base address and size; • the value written is the same as the previously written doorbell value; • the number of commands that would be added as part of a doorbell write would exceed the number of available entries; • host software attempts to add a command to a full Submission Queue; and • host software attempts to remove a completion queue entry from an empty Completion Queue.
02h	**Diagnostic Failure:** A diagnostic failure was detected. This may include a self-test operation.
03h	**Persistent Internal Error:** A failure occurred that is persistent and the controller is unable to isolate to a specific set of commands. If this error is indicated, then the CSTS.CFS bit may be set to '1' and the host should perform a reset as described in section 3.6.
04h	**Transient Internal Error:** A transient error occurred that is specific to a particular set of commands; controller operation may continue without a reset.
05h	**Firmware Image Load Error:** The firmware image could not be loaded. The controller reverted to the previously active firmware image or a baseline read-only firmware image.
06h to FFh	Reserved

Figure 146: Asynchronous Event Information – SMART / Health Status

Value	Description
00h	**NVM subsystem Reliability:** NVM subsystem reliability has been compromised. This may be due to significant media errors, an internal error, the media being placed in read only mode, or a volatile memory backup device failing. This status value shall not be used if the read-only condition on the media is due to a change in the write protection state of a namespace (refer to section 8.12.1).
01h	**Temperature Threshold:** A temperature is greater than or equal to an over temperature threshold or less than or equal to an under temperature threshold (refer to section 5.27.1.3).
02h	**Spare Below Threshold:** Available spare capacity has fallen below the threshold.
03h to FFh	Reserved

Figure 147: Asynchronous Event Information – Notice

Value	Description
00h	**Namespace Attribute Changed:** Indicates a change to one or more of the following: • the Identify Namespace data structure (refer to the applicable NVMe I/O Command Set specification) for one or more namespaces; • the I/O Command Set Independent Identify Namespace data structure; • the Namespace List returned when the Identify command is issued with the CNS field set to 02h; or • other data structures as specified in applicable NVMe I/O Command Set specifications. To clear this event, host software issues a Get Log Page command for the Changed Namespace List log page (refer to section 5.16.1.5) with the Retain Asynchronous Event bit cleared to '0'. A controller shall not send this event if: a) Namespace Status (refer to Figure 280) has changed and shutdown processing is either occurring (i.e., CSTS.SHST is set to 01b) or complete (i.e., CSTS.SHST is set to 10b); b) the ANAGRPID field (refer to Figure 280) has changed; or c) an I/O Command Set specific change occurs (refer to the applicable I/O Command Set specification). A controller shall only send this event for changes to the Format Progress Indicator field when bits 6:0 of that field transition from a non-zero value to 0h, or from 0h to a non-zero value.
01h	**Firmware Activation Starting:** The controller is starting a firmware activation process during which command processing is paused. Host software may use CSTS.PP to determine when command processing has resumed. To clear this event, host software reads the Firmware Slot Information log page.
02h	**Telemetry Log Changed:** The controller has saved the controller internal state in the Telemetry Controller-Initiated log page and set the Telemetry Controller-Initiated Data Available field to 1h in that log page. To clear this event, the host issues a Get Log Page command with Retain Asynchronous Event bit cleared to '0' for the Telemetry Controller-Initiated log.
03h	**Asymmetric Namespace Access Change:** The Asymmetric Namespace Access information (refer to section 5.16.1.13) related to an ANA Group that contains namespaces attached to this controller has changed (e.g., an ANA state has changed, an ANAGRPID has changed). The current Asymmetric Namespace Access information for attached namespaces is indicated in the Asymmetric Namespace Access log page (refer to section 5.16.1.13). To clear this event, the host issues a Get Log Page command (refer to section 5.16) with the Retain Asynchronous Event bit cleared to '0' for the Asymmetric Namespace Access log. A controller shall not send this event if a Namespace Attribute Changed notice is sent for the same event, such as a change due to: a) the attachment of a namespace (refer to section 5.22); b) the deletion of a namespace (refer to section 5.23); or c) the detachment of a namespace (refer to section 5.22).
04h	**Predictable Latency Event Aggregate Log Change:** Indicates that event pending entries for one or more NVM Sets (refer to section 5.16.1.12) have been added to the Predictable Latency Event Aggregate log.

Figure 147: Asynchronous Event Information – Notice

Value	Description
05h [1]	**LBA Status Information Alert:** I/O Command Set specific definition.
06h	**Endurance Group Event Aggregate Log Page Change:** Indicates that event entries for one or more Endurance Groups (refer to section 5.16.1.10) have been added to the Endurance Group Event Aggregate log. To clear this event, the host issues a Get Log Page command with the Retain Asynchronous Event bit cleared to '0' for the Endurance Group Event Aggregate log.
07h to EEh	Reserved
EFh [2]	**Zone Descriptor Changed:** I/O Command Set specific definition.
F0h	**Discovery Log Page Change:** A change has occurred to one or more of the Discovery Log Pages. The host should submit a Get Log Page command to receive updated Discovery Log Pages.
F1h to FFh	Reserved for future NVMe over Fabrics Asynchronous Event Notifications
NOTE: 1. Refer to the NVM Command Set Specification. 2. Refer to the Zoned Namespace Command Set Specification.	

Figure 148: Asynchronous Event Information – I/O Command Specific Status

Value	Description
00h	**Reservation Log Page Available:** Indicates that one or more Reservation Notification log pages (refer to section 5.16.1.24) have been added to the Reservation Notification log.
01h	**Sanitize Operation Completed:** Indicates that a sanitize operation has completed (including any associated additional media modification, refer to the No-Deallocate Modifies Media After Sanitize field in Figure 275) without unexpected deallocation of all user data (refer to section 5.27.1.19) and status is available in the Sanitize Status log page (refer to section 5.16.1.25).
02h	**Sanitize Operation Completed With Unexpected Deallocation:** Indicates that a sanitize operation for which No-Deallocate After Sanitize (refer to Figure 303) was requested has completed with the unexpected deallocation of all user data (refer to section 5.27.1.19) and status is available in the Sanitize Status log page (refer to section 5.16.1.25).
03h to FFh	Reserved

Figure 149: Asynchronous Event Information – Immediate

Value	Description
00h	**NVM Subsystem Normal Shutdown:** This controller has started performing a normal NVM Subsystem Shutdown that is due to: • the value 4E726D6Ch ("Nrml") has been written to an NSS.NCCR register within the NVM subsystem or Domain; or • an NVMe-MI Shutdown command (refer to the NVM Express Management Interface Specification) being processed. Refer to section 3.6.3.
01h to FFh	Reserved

5.3 Capacity Management command

Host software uses the Capacity Management command to configure Endurance Groups and NVM Sets in an NVM subsystem by either selecting one of a set of supported configurations (i.e., Fixed Capacity Management; refer to section 8.3.2) or by specifying the capacity of the Endurance Group or NVM Set to be created (i.e., Variable Capacity Management; refer to section 8.3.3). The Capacity Management command specifies the operations defined in Figure 150.

The Capacity Management command uses the Command Dword 10, Command Dword 11, and Command Dword 12 fields. All other command specific fields are reserved.

For requirements to support the operations in Figure 150, refer to section 8.3.

Figure 150: Capacity Management – Command Dword 10

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:16</td><td>Element Identifier: This field contains a value specific to the value of the Operation field.[1]</td></tr>
<tr><td>15:04</td><td>Reserved</td></tr>
<tr><td>03:00</td><td>Operation: Specifies the operation to be performed by the controller:
<table>
<tr><th>Value</th><th>Description</th><th>Element Identifier Field</th></tr>
<tr><td>0h</td><td>Select Capacity Configuration: Endurance Groups and NVM Sets are configured as indicated by the Capacity Configuration Descriptor specified by this command (refer to section 5.3.1).</td><td>Capacity Configuration Identifier (refer to Figure 251).</td></tr>
<tr><td>1h</td><td>Create Endurance Group: An Endurance Group is created (refer to section 8.3.3) with the capacity specified by the Capacity Lower field (refer to Figure 151) and the Capacity Upper field (refer to Figure 152).</td><td>Domain Identifier (refer to section 3.2.4.3) of the domain in which the Endurance Group is to be created. A value of 0h specifies that the controller selects the domain in which the Endurance Group is created.</td></tr>
<tr><td>2h</td><td>Delete Endurance Group: The Endurance Group specified by this command is deleted (refer to section 8.3.3). All namespaces and NVM Sets contained by the Endurance Group are deleted.</td><td>Endurance Group Identifier of the Endurance Group to be deleted.</td></tr>
<tr><td>3h</td><td>Create NVM Set: An NVM Set is created (refer to section 8.3.3) in the specified Endurance Group with the capacity specified by the Capacity Lower field and the Capacity Upper field.
If the controller does not support NVM Sets, then this operation is not supported.</td><td>Endurance Group Identifier of the Endurance Group in which the NVM Set is to be created. A value of 0h specifies that the controller selects the Endurance Group in which the NVM Set is created.</td></tr>
<tr><td>4h</td><td>Delete NVM Set: The NVM Set specified by this command is deleted (refer to section 8.3.3). All namespaces in the NVM Set are deleted.
If the controller does not support NVM Sets, then this operation is not supported.</td><td>NVM Set identifier of the NVM Set to be deleted.</td></tr>
<tr><td>5h to Fh</td><td>Reserved</td><td></td></tr>
</table></td></tr>
</table>

Notes:
1. If the Element Identifier field specifies a non-zero value which does not correspond to an existing capacity entity, then the controller shall abort the command with a status code of Invalid Field in Command.

Figure 151: Capacity Management – Command Dword 11

Bits	Description
31:00	**Capacity Lower:** This field specifies the least significant 32 bits of the capacity in bytes of the Endurance Group or NVM Set to be created.

Figure 152: Capacity Management – Command Dword 12

Bits	Description
31:00	**Capacity Upper:** This field specifies the most significant 32 bits of the capacity in bytes of the Endurance Group or NVM Set to be created.

If the Operation field specifies the Create Endurance Group operation or the Create NVM Set operation, then the Capacity Upper and Capacity Lower fields specify the capacity of the Endurance Group or NVM Set to be created. If the Operation field specifies any other operation, then the Capacity Upper field and the Capacity Lower field are reserved.

5.3.1 Media Unit Configuration Selection

If:

a) the Operation field specifies the Select Capacity Configuration operation;
b) the Element Identifier field specifies a Capacity Configuration Descriptor in the Supported Capacity Configuration List; and
c) the Selected Configuration field of the Media Unit Status log page (refer to Figure 248) is cleared to 0h,

then the controller shall perform all of the following actions in sequence:

1) Create an Endurance Group for each Endurance Group Configuration Descriptor in the selected Capacity Configuration Descriptor.
2) Create an NVM Set for each NVM Set Identifier specified in each Endurance Group Configuration Descriptor, if any.

If the Operation field specifies the Select Capacity Configuration operation and the Element Identifier field is cleared to '0', then the controller shall clear the configuration by performing all of the following actions in sequence:

1) Delete all namespaces in the domain that contains the controller processing the command, as described in section 8.11.
2) Delete all NVM Sets in the domain that contains the controller processing the command, if any.
3) Delete all Endurance Groups in the domain that contains the controller processing the command.
4) Clear the Selected Configuration field to 0h in the Media Unit Status log page.

If the Operation field specifies the Select Capacity Configuration operation and the Element Identifier field specifies the value reported in the Selected Configuration field of the Media Unit Status log page (i.e., the currently-selected configuration), then the controller shall complete the command without error and shall make no change to the capacity configuration.

If the Operation field specifies the Select Capacity Configuration operation and:

a) the Element Identifier field does not specify either a value of 0h or the Capacity Configuration Identifier of a Capacity Configuration Descriptor in the Capacity Configuration List; or
b) the Selected Configuration field of the Media Unit Status log page (refer to Figure 248) is not cleared to 0h,

then the controller shall abort the command with a status code of Invalid Field in Command and no changes shall be made to the configuration of any Media Unit.

5.3.2 Endurance Group Operations

If the Operation field specifies the Create Endurance Group operation, the controller shall select a non-zero Endurance Group Identifier not assigned to an existing Endurance Group in the specified domain (refer to Figure 150) and indicate that value in Dword 0 of the completion queue entry (refer to Figure 154). If a non-zero unassigned Endurance Group Identier is unavailable, then the controller shall abort the command with a status code of Identifier Unavailable.

If the Operation field specifies the Create Endurance Group operation and Media Units are supported, then the controller selects Media Units from the specified domain to be allocated to the Endurance Group.

If the Operation field specifies the Create Endurance Group operation and Media Units are not supported, then the controller selects NVM capacity from the specified domain to be allocated to the Endurance Group.

If the Operation field specifies the Create Endurance Group operation and the Capacity Lower and Capacity Upper fields specify a value that requires allocation of NVM capacity that is greater than the value in:

a) the Max Endurance Group Capacity (MEGCAP) field in the Identify Controller data structure;
b) the Unallocated NVM Capacity (UNVMCAP) field in the Identify Controller data structure; or
c) the Max Endurance Group Capacity (MEGCAP) field in the Domain Attributes Entry for the domain in which the Endurance Group is being created,

then the controller:

a) shall abort the command with Insufficient Capacity status; and
b) if the Error Information log page is supported, shall indicate in the Command Specific Information field the total amount of NVM capacity in bytes of the largest Endurance Group that is able to be created.

If the Operation field specifies the Delete Endurance Group operation and the Element Identifier field specifies a value of 0h or the identifier of an Endurance Group that does not exist, then the controller shall abort the command with Invalid Field in Command status.

5.3.3 NVM Set Operations

If the Operation field specifies the Create NVM Set operation, the controller shall select a non-zero NVM Set Identifier not assigned to an existing NVM Set in the specified Endurance Group (refer to Figure 150) and indicate that value in the completion queue entry. If a non-zero unassigned NVM Set Identifier is unavailable, then the controller shall abort the command with a status code of Identifier Unavailable.

If the Operation field specifies the Create NVM Set operation and the Capacity Lower and Capacity Upper fields specify a value that requires allocation of NVM capacity that is greater than the value in the Unallocated Endurance Group Capacity (UEGCAP) field in the Endurance Group Information log page (refer to Figure 217) for the specified Endurance Group, then the controller:

a) if the Error Information log page is supported, shall indicate in the Command Specific Information field the total amount of NVM capacity in bytes of the largest NVM Set that is able to be created; and
b) shall abort the command with Insufficient Capacity status.

If the Operation field specifies the Create NVM Set operation and the Element Identifier field is cleared to 0h, then the controller shall choose an existing Endurance Group in an existing domain in which to create the NVM Set.

If the Operation field specifies the Create NVM Set operation and Media Units are supported, then the controller selects Media Units from the Endurance Group to be allocated to the NVM Set.

If the Operation field specifies the Create NVM Set operation and Media Units are not supported, then the controller selects NVM capacity from the Endurance Group to be allocated to the NVM Set.

If the Operation field specifies the Delete NVM Set operation and the Element Identifier field specifies a value of 0h or the identifier of an NVM Set that does not exist, then the controller shall abort the command with Invalid Field in Command status.

If the controller does not support NVM Sets and the Operation field specifies either the Create NVM Set operation or the Delete NVM Set operation, then the controller shall abort the command with Invalid Field in Command status.

5.3.4 Command Completion

Upon completion of the Capacity Management command, the controller posts a completion queue entry to the Admin Completion Queue. Capacity Management command specific status values are defined in Figure 153.

Figure 153: Capacity Management – Command Specific Status Values

Value	Description
26h	**Insufficient Capacity:** The requested operation requires more free space than is currently available. The Command Specific Information field of the Error Information log page (refer to Figure 193) specifies the total amount of NVM capacity in bytes required to create the Endurance Group or NVM Set.
2Dh	**Identifer Unavailable:** The number of Endurance Groups or NVM Sets supported has been exceeded.

Dword 0 of the completion queue entry contains the identifier of the Endurance Group or NVM Set created, if any. Completion queue entry Dword 0 is defined in Figure 154.

Figure 154: Capacity Management – Completion Queue Entry Dword 0

Bits	Description
31:16	Reserved
15:00	**Created Element Identifier:** This field indicates the identifier of the NVM Set that was created if the Create NVM Set operation was used. This field indicates the identifier of the Endurance Group Identifier if the Create Endurance Group operation was used. This field is reserved for all other operations.

5.4 Create I/O Completion Queue command

The Create I/O Completion Queue command is used to create all I/O Completion Queues with the exception of the Admin Completion Queue. The Admin Completion Queue is created by specifying its base address in the ACQ property. If a PRP List is provided to describe the CQ, then the PRP List shall be maintained by host software at the same location in host physical memory and the values in the PRP List shall not be modified until the corresponding Delete I/O Completion Queue command for this CQ is completed successfully or the controller is reset. If the PRP List values are modified, the behavior is undefined.

The Create I/O Completion Queue command uses the PRP Entry 1, Command Dword 10, and Command Dword 11 fields. All other command specific fields are reserved.

Figure 155: Create I/O Completion Queue – PRP Entry 1

Bits	Description
63:00	**PRP Entry 1 (PRP1):** If CDW11.PC is set to '1', then this field specifies a 64-bit base memory address pointer of the Completion Queue that is physically contiguous. The address pointer is memory page aligned (based on the value in CC.MPS) unless otherwise specified. If CDW11.PC is cleared to '0', then this field specifies a PRP List pointer that describes the list of pages that constitute the Completion Queue. The list of pages is memory page aligned (based on the value in CC.MPS) unless otherwise specified. In both cases the PRP Entry shall have an offset of 0h. In a non-contiguous Completion Queue, each PRP Entry in the PRP List shall have an offset of 0h. If there is a PRP Entry with a non-zero offset, then the controller should return an error of PRP Offset Invalid.

Figure 156: Create I/O Completion Queue – Command Dword 10

Bits	Description
31:16	**Queue Size (QSIZE):** This field indicates the size of the Completion Queue to be created. If the size is 0h or larger than the controller supports, the controller should return an error of Invalid Queue Size. Refer to section 3.3.3.2.2. This is a 0's based value.
15:00	**Queue Identifier (QID):** This field indicates the identifier to assign to the Completion Queue to be created. This identifier corresponds to the Completion Queue Head Doorbell used for this command (i.e., the value *y* in the CQyHDBL section of the NVMe over PCIe Transport Specification). This value shall not exceed the value reported in the Number of Queues feature (refer to section 5.27.1.5) for I/O Completion Queues. If the value specified is 0h, exceeds the Number of Queues reported, or corresponds to an identifier already in use, the controller should return an error of Invalid Queue Identifier.

Figure 157: Create I/O Completion Queue – Command Dword 11

Bits	Description
31:16	**Interrupt Vector (IV):** This field's value is transport specific and is described in the applicable NVMe Transport binding specification if required. If not defined by the transport, then this field shall be cleared to 0h.
15:02	Reserved

Figure 157: Create I/O Completion Queue – Command Dword 11

Bits	Description
01	**Interrupts Enabled (IEN):** If set to '1', then interrupts are enabled for this Completion Queue. If cleared to '0', then interrupts are disabled for this Completion Queue.
00	**Physically Contiguous (PC):** If set to '1', then the Completion Queue is physically contiguous and PRP Entry 1 (PRP1) is the address of a contiguous physical buffer. If cleared to '0', then the Completion Queue is not physically contiguous and PRP Entry 1 (PRP1) is a PRP List pointer. If this bit is cleared to '0' and CAP.CQR is set to '1', then the controller should return an error of Invalid Field in Command. If the: • queue is located in the Controller Memory Buffer; • PC is cleared to '0'; and • CMBLOC.CQPDS is cleared to '0', then the controller shall abort the command with Invalid Use of Controller Memory Buffer status.

5.4.1 Command Completion

If the command is completed, then the controller shall post a completion queue entry to the Admin Completion Queue indicating the status for the command.

Create I/O Completion Queue command specific status values are defined in Figure 158.

Figure 158: Create I/O Completion Queue – Command Specific Status Values

Value	Description
1h	**Invalid Queue Identifier:** The creation of the I/O Completion Queue failed due to an invalid queue identifier specified as part of the command. An invalid queue identifier is one that identifies the Admin Queue (i.e., 0h), is outside the range supported by the controller, or is a Completion Queue Identifier that is already in use.
2h	**Invalid Queue Size:** The host attempted to create an I/O Completion Queue: • with an invalid number of entries (e.g., a value of 0h or a value which exceeds the maximum supported by the controller, specified in CAP.MQES); or • before initializing the CC.IOCQES field.
8h	**Invalid Interrupt Vector:** The creation of the I/O Completion Queue failed due to an invalid interrupt vector specified as part of the command.

5.5 Create I/O Submission Queue command

The Create I/O Submission Queue command is used to create I/O Submission Queues. The Admin Submission Queue is created by specifying its base address in the ASQ property. If a PRP List is provided to describe the SQ to be created, then the PRP List shall be maintained by host software at the same location in host physical memory and the values in the PRP List shall not be modified until the corresponding Delete I/O Submission Queue command for that SQ is completed or the controller is reset. If the PRP List values are modified, the behavior is undefined.

The Create I/O Submission Queue command uses the PRP Entry 1, Command Dword 10, Command Dword 11, and Command Dword 12 fields. All other command specific fields are reserved.

Figure 159: Create I/O Submission Queue – PRP Entry 1

Bits	Description
63:00	**PRP Entry 1 (PRP1):** If CDW11.PC is set to '1', then this field specifies a 64-bit base memory address pointer of the Submission Queue that is physically contiguous. The address pointer is memory page aligned (based on the value in CC.MPS) unless otherwise specified. If CDW11.PC is cleared to '0', then this field specifies a PRP List pointer that describes the list of pages that constitute the Submission Queue. The list of pages is memory page aligned (based on the value in CC.MPS) unless otherwise specified. In both cases, the PRP Entry shall have an offset of 0h. In a non-contiguous Submission Queue, each PRP Entry in the PRP List shall have an offset of 0h. If there is a PRP Entry with a non-zero offset, then the controller should return an error of PRP Offset Invalid.

Figure 160: Create I/O Submission Queue – Command Dword 10

Bits	Description
31:16	**Queue Size (QSIZE):** This field indicates the size of the Submission Queue to be created. If the size is 0h or larger than the controller supports, the controller should return an error of Invalid Queue Size. Refer to section 3.3.3.2.2. This is a 0's based value.
15:00	**Queue Identifier (QID):** This field indicates the identifier to assign to the Submission Queue to be created. This identifier corresponds to the Submission Queue Tail Doorbell used for this command (i.e., the value *y* in SQyTDBL section of the NVMe over PCIe Transport Specification). This value shall not exceed the value reported in the Number of Queues feature (refer to section 5.27.1.5) for I/O Submission Queues. If the value specified is 0h, exceeds the Number of Queues reported, or corresponds to an identifier already in use, the controller should return an error of Invalid Queue Identifier.

Figure 161: Create I/O Submission Queue – Command Dword 11

Bits	Description
31:16	**Completion Queue Identifier (CQID):** This field indicates the identifier of the I/O Completion Queue to utilize for any command completions entries associated with this Submission Queue. If the value specified: a) is 0h (i.e., the Admin Completion Queue), then the controller should return an error of Invalid Queue Identifier; b) is outside the range supported by the controller, then the controller should return an error of Invalid Queue Identifier; or c) is within the range supported by the controller and does not identify an I/O Completion Queue that has been created, then the controller should return an error of Completion Queue Invalid.
15:03	Reserved
02:01	**Queue Priority (QPRIO):** This field indicates the priority class to use for commands within this Submission Queue. This field is only used when the weighted round robin with urgent priority class is the arbitration mechanism selected, the field is ignored if weighted round robin with urgent priority class is not used. Refer to section 3.4.4. **Value** / **Definition** 00b – Urgent 01b – High 10b – Medium 11b – Low

Figure 161: Create I/O Submission Queue – Command Dword 11

Bits	Description
00	**Physically Contiguous (PC):** If set to '1', then the Submission Queue is physically contiguous and PRP Entry 1 (PRP1) is the address of a contiguous physical buffer. If cleared to '0', then the Submission Queue is not physically contiguous and PRP Entry 1 (PRP1) is a PRP List pointer. If this bit is cleared to '0' and CAP.CQR is set to '1', then the controller should return an error of Invalid Field in Command. If the: • queue is located in the Controller Memory Buffer; • PC is cleared to '0'; and • CMBLOC.CQPDS is cleared to '0', then the controller shall abort the command with Invalid Use of Controller Memory Buffer status.

Figure 162: Create I/O Submission Queue – Command Dword 12

Bits	Description
31:16	Reserved
15:00	**NVM Set Identifier (NVMSETID):** This field indicates the identifier of the NVM Set to be associated with this Submission Queue. If this field is cleared to 0h or the SQ Associations capability is not supported (refer to section 8.22), then this Submission Queue is not associated with any specific NVM Set. If this field is set to a non-zero value that is not specified in the NVM Set List (refer to Figure 278) and the SQ Associations feature is supported (refer to section 8.22), then the controller shall abort the command with a status code of Invalid Field in Command. The host should not submit commands for namespaces associated with other NVM Sets in this Submission Queue (refer to section 8.22).

5.5.1 Command Completion

Upon completion of the Create I/O Submission Queue command, the controller posts a completion queue entry to the Admin Completion Queue.

Create I/O Submission Queue command specific status values are defined in Figure 163.

Figure 163: Create I/O Submission Queue – Command Specific Status Values

Value	Description
0h	**Completion Queue Invalid:** The Completion Queue identifier specified in the command has not been created.
1h	**Invalid Queue Identifier:** The creation of the I/O Submission Queue failed due an invalid queue identifier specified as part of the command. An invalid queue identifier is one that identifies the Admin Queue (i.e., 0h), is outside the range supported by the controller, or is a Submission Queue Identifier that is already in use.
2h	**Invalid Queue Size:** The host attempted to create an I/O Submission Queue: • with an invalid number of entries (e.g., a value of 0h or a value which exceeds the maximum supported by the controller, specified in CAP.MQES); or • before initializing the CC.IOSQES field.

5.6 Delete I/O Completion Queue command

The Delete I/O Completion Queue command is used to delete an I/O Completion Queue. The Delete I/O Completion Queue command uses the Command Dword 10 field. All other command specific fields are reserved. After this command has completed, the PRP List that describes the Completion Queue may be deallocated by host software.

Host software shall ensure that any associated I/O Submission Queue is deleted prior to deleting a Completion Queue. If there are any associated I/O Submission Queues present, then the Delete I/O Completion Queue command shall abort with a status code of Invalid Queue Deletion.

Note: It is not possible to delete the Admin Completion Queue.

Figure 164: Delete I/O Completion Queue – Command Dword 10

Bits	Description
31:16	Reserved
15:00	**Queue Identifier (QID):** This field indicates the identifier of the Completion Queue to be deleted. The value of 0h (Admin Completion Queue) shall not be specified.

5.6.1 Command Completion

Upon completion of the Delete I/O command, the controller posts a completion queue entry to the Admin Completion Queue. Delete I/O Completion Queue command specific status values are defined in Figure 165.

Figure 165: Delete I/O Completion Queue – Command Specific Status Values

Value	Description
01h	**Invalid Queue Identifier:** The Queue Identifier specified in the command is invalid. This error is also indicated if the Admin Completion Queue identifier is specified.
0Ch	**Invalid Queue Deletion:** This error indicates that it is invalid to delete the I/O Completion Queue specified. The typical reason for this error condition is that there is an associated I/O Submission Queue that has not been deleted.

5.7 Delete I/O Submission Queue command

The Delete I/O Submission Queue command is used to delete an I/O Submission Queue. The Delete I/O Submission Queue command uses the Command Dword 10 field. All other command specific fields are reserved. After this command has completed, the PRP List that describes the Submission Queue may be deallocated by host software.

Upon successful completion of the Delete I/O Submission Queue command, all I/O commands previously submitted to the indicated Submission Queue shall be either explicitly completed or implicitly completed. Prior to returning a completion queue entry for the Delete I/O Submission Queue command, other commands previously submitted to the I/O Submission Queue to be deleted may be completed with appropriate status (e.g., Successful Completion, Command Aborted due to SQ Deletion). After successful completion of the Delete I/O Submission Queue command, the controller shall not post completion status for any I/O commands that were submitted to the deleted I/O Submission Queue. The successful completion of the Delete I/O Submission Queue command indicates an implicit completion status of Command Aborted due to SQ Deletion for any previously submitted I/O commands that did not have a completion queue entry posted by the controller.

Note: It is not possible to delete the Admin Submission Queue.

Figure 166: Delete I/O Submission Queue – Command Dword 10

Bits	Description
31:16	Reserved
15:00	**Queue Identifier (QID):** This field indicates the identifier of the Submission Queue to be deleted. The value of 0h (Admin Submission Queue) shall not be specified.

5.7.1 Command Completion

After all commands submitted to the indicated I/O Submission Queue are either completed or aborted, a completion queue entry is posted to the Admin Completion Queue when the queue has been deleted. Delete I/O Submission Queue command specific status values are defined in Figure 167.

Figure 167: Delete I/O Submission Queue – Command Specific Status Values

Value	Description
1h	**Invalid Queue Identifier:** The Queue Identifier specified in the command is invalid. This error is also indicated if the Admin Submission Queue identifier is specified.

5.8 Doorbell Buffer Config command

The Doorbell Buffer Config command is used to provide two separate memory buffers that mirror the controller's doorbell properties defined in section 3.1.3. This command is intended for emulated controllers and is not typically supported by a physical NVMe controller. The two buffers are known as “Shadow Doorbell” and “EventIdx”, respectively. Refer to “Updating Controller Doorbell Properties using a Shadow Doorbell Buffer” in the annex for an example of how these buffers may be used.

The Doorbell Buffer Config command uses the PRP Entry 1 and PRP Entry 2 fields. All other command specific fields are reserved. The command is not namespace specific, does not support metadata, and does not support SGLs. The settings are not retained across a Controller Level Reset.

Each buffer supplied with the Doorbell Buffer Config command shall be a single physical memory page as defined by the CC.MPS field. The controller shall ensure that the following condition is satisfied:

(4 << CAP.DSTRD) * (max(NSQA, NCQA)+1) <= (2^(12+CC.MPS))

Figure 168: Doorbell Buffer Config – Shadow Doorbell and EventIdx

Start (Offset in Buffer)[1, 2]	End (Offset in Buffer)[1, 2]	Description [2]
00h	03h	Submission Queue 0 Tail Doorbell or EventIdx (Admin)
00h + (1 * (4 << CAP.DSTRD))	03h + (1 * (4 << CAP.DSTRD))	Completion Queue 0 Head Doorbell or EventIdx (Admin)
00h + (2 * (4 << CAP.DSTRD))	03h + (2 * (4 << CAP.DSTRD))	Submission Queue 1 Tail Doorbell or EventIdx
00h + (3 * (4 << CAP.DSTRD))	03h + (3 * (4 << CAP.DSTRD))	Completion Queue 1 Head Doorbell or EventIdx
…	…	...
00h + (2y * (4 << CAP.DSTRD))	03h + (2y * (4 << CAP.DSTRD))	Submission Queue y Tail Doorbell or EventIdx
00h + ((2y + 1) * (4 << CAP.DSTRD))	03h + ((2y + 1) * (4 << CAP.DSTRD))	Completion Queue y Head Doorbell or EventIdx

Notes:
1. The offsets in Start and End are referenced to the value provided in PRP1 for the doorbell buffer and to the value provided in PRP2 for the EventIdx buffer.
2. The value of y is equal to max(NSQA, NCQA).

Figure 169: Doorbell Buffer Config – PRP Entry 1

Bits	Description
63:00	**PRP Entry 1 (PRP1):** This field specifies a 64-bit base memory address pointer to the Shadow Doorbell buffer with the definition specified in Figure 168. The Shadow Doorbell buffer is updated by the host. This buffer shall be memory page aligned.

Figure 170: Doorbell Buffer Config – PRP Entry 2

Bits	Description
63:00	**PRP Entry 2 (PRP2):** This field specifies a 64-bit base memory address pointer to the EventIdx buffer with the definition specified in Figure 168. The EventIdx buffer is updated by the para-virtualized controller. This buffer shall be memory page aligned.

5.8.1 Command Completion

When the command is completed, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command. If the Shadow Doorbell buffer or EventIdx buffer memory addresses are invalid, then a status code of Invalid Field in Command shall be returned.

5.9 Device Self-test command

The Device Self-test command is used to start a device self-test operation or abort a device self-test operation (refer to section 8.6). The Device Self-test command is used specifically to:

a) start a short device self-test operation;
b) start an extended device self-test operation;
c) start a vendor specific device self-test operation; or
d) abort a device self-test operation already in process.

The device self-test operation is performed by the controller that the Device Self-test command was submitted to. The Namespace Identifier field controls which namespaces are included in the device self-test operation as specified in Figure 171.

Figure 171: Device Self-test Namespace Test Action

NSID Value	Description
00000000h	Specifies that the device self-test operation shall not include any namespaces, and only the controller is included as part of the device self-test operation.
00000001h to FFFFFFFEh	Specifies that the device self-test operation shall include the namespace specified by this field. If this field specifies an invalid namespace ID, then the controller shall abort the command with a status code of Invalid Namespace or Format. If this field specifies an inactive namespace ID, then the controller shall abort the command with a status code of Invalid Field in Command.
FFFFFFFFh	Specifies that the device self-test operation shall include all active namespaces accessible through the controller at the time the device self-test operation is started.

The Device Self-test command uses the Command Dword 10 field. All other command specific fields are reserved.

Figure 172: Device Self-test – Command Dword 10

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:04</td><td>Reserved</td></tr>
<tr><td>03:00</td><td>Self-test Code (STC): This field specifies the action taken by the Device Self-test command.

<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0h</td><td>Reserved</td></tr>
<tr><td>1h</td><td>Start a short device self-test operation</td></tr>
<tr><td>2h</td><td>Start an extended device self-test operation</td></tr>
<tr><td>3h to Dh</td><td>Reserved</td></tr>
<tr><td>Eh</td><td>Vendor specific</td></tr>
<tr><td>Fh</td><td>Abort device self-test operation</td></tr>
</table>
</td></tr>
</table>

The processing of a Device Self-test command and interactions with a device self-test operation already in progress is defined in Figure 173.

Figure 173: Device Self-test – Command Processing

Self-test in Progress [1]	Self-test Code value in new Drive Self-test command	Controller Action
Yes	1h – Short device self-test	Abort the new Device Self-test command with a status code of Device Self-test in Progress.
	2h – Extended device self-test	
	Eh – Vendor specific	Vendor specific
	Fh – Abort device self-test	The controller takes the following actions in order: 1. Abort device self-test operation in progress; 2. Create log entry in the Newest Self-test Result Data Structure in the Device Self-test Log; 3. Set the Current Device Self-test Status field in the Device Self-test Log to 0h; and 4. Completes command successfully.
No	1h – Short device self-test	The controller takes the following actions in order: 1. Validate the command parameters; 2. Set the Current Device Self-test Status field in the Device Self-test Log to 1h; 3. Start a device self-test operation; and 4. Completes command successfully.
	2h – Extended device self-test	The controller takes the following actions in order: 1. Validate the command parameters; 2. Set the Current Device Self-test Status field in the Device Self-test Log to 2h; 3. Start a device self-test operation; and 4. Completes command successfully.
	Eh – Vendor specific	Vendor specific
	Fh – Abort device self-test	Completes command successfully. The Device Self-test Log is not modified.

Notes:
1. If bit 0 is cleared to '0' in the Device Self-test Options (DSTO) of the Identify Controller data structure (refer to Figure 275), then the Self-test in Progress column represents that a device self-test operation is in progress on the controller that the new Device Self-test command was received on. If bit 0 is set to '1' in the Device Self-test Options (DSTO) of the Identify Controller data structure, then the Self-test in Progress column represents that a device self-test operation is in progress on the NVM subsystem.

5.9.1 Command Completion

A completion queue entry is posted to the Admin Completion Queue after the appropriate actions are taken as specified in Figure 173. Device Self-test command specific status values are defined in Figure 174.

Figure 174: Device Self-test – Command Specific Status Values

Value	Description
1Dh	**Device Self-test in Progress:** The controller or NVM subsystem already has a device self-test operation in process.

5.10 Directive Receive command

The Directive Receive command returns a data buffer that is dependent on the Directive Type. Refer to section 8.7.

The Directive Receive command uses the Data Pointer, Command Dword 10, and Command Dword 11 fields. Command Dword 12 and Dword 13 may be used based on the Directive Type field and the Directive Operation field. All other command specific fields are reserved.

If the Number of Dwords (NUMD) field corresponds to a length that is less than the size of the data structure to be returned, then only that specified portion of the data structure is transferred. If the NUMD field corresponds to a length that is greater than the size of the associated data structure, then the entire contents of the data structure are transferred and no additional data is transferred.

Figure 175: Directive Receive – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 87 for the definition of this field.

Figure 176: Directive Receive – Command Dword 10

Bits	Description
31:00	**Number of Dwords (NUMD):** This field specifies the number of dwords to transfer. This is a 0's based value.

Figure 177: Directive Receive – Command Dword 11

Bits	Description
31:16	**Directive Specific (DSPEC):** The interpretation of this field is Directive Type dependent. Refer to section 8.7.
15:08	**Directive Type (DTYPE):** This field specifies the Directive Type. Refer to Figure 416 for the list of Directive Types.
07:00	**Directive Operation (DOPER):** This field specifies the Directive Operation to perform. The interpretation of this field is Directive Type dependent. Refer to section 8.7.

5.10.1 Command Completion

When the command is completed, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command. Command specific status values that may be returned are dependent on the Directive Type, refer to section 8.7.

5.11 Directive Send command

The Directive Send command transfers a data buffer that is dependent on the Directive Type to the controller. Refer to section 8.7.

The Directive Send command uses the Data Pointer, Command Dword 10, and Command Dword 11 fields. Command Dword 12 and Command Dword 13 may be used based on the Directive Type field and the Directive Operation field. All other command specific fields are reserved.

Figure 178: Directive Send – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 87 for the definition of this field.

Figure 179: Directive Send – Command Dword 10

Bits	Description
31:00	**Number of Dwords (NUMD):** This field specifies the number of dwords to transfer. This is a 0's based value.

Figure 180: Directive Send – Command Dword 11

Bits	Description
31:16	**Directive Specific (DSPEC):** The interpretation of this field is Directive Type dependent. Refer to section 8.7.
15:08	**Directive Type (DTYPE):** This field specifies the Directive Type. Refer to Figure 416 for the list of Directive Types.
07:00	**Directive Operation (DOPER):** This field specifies the Directive Operation to perform. The interpretation of this field is Directive Type dependent. Refer to section 8.7.

5.11.1 Command Completion

When the command is completed, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command. Command specific status values that may be returned are dependent on the Directive Type, refer to section 8.7.

5.12 Firmware Commit command

Note: This command was known in NVM Express Base Specification revisions prior to revision 1.2 as “Firmware Activate.”

The Firmware Commit command is used to modify the firmware image or Boot Partitions.

When modifying a firmware image, the Firmware Commit command verifies that a valid firmware image has been downloaded and commits that revision to a specific firmware slot. The host may select the firmware image to activate on the next Controller Level Reset as part of this command. The host may determine the currently executing firmware revision by examining the Firmware Revision field in the Identify Controller data structure in Figure 275. The host may determine the firmware revision to be executed on the next Controller Level Reset by examining the Firmware Slot Information log page. All controllers in a domain share firmware slots and the same firmware image is applied to all controllers in that domain (i.e., all the controllers in the NVM subsystem if multiple domains are not supported or all the controllers in that domain if multiple domains are supported).

Activation of a firmware image may result in a change in controller behavior that is not expected by the host (e.g., an incompatible change in the UUID List (refer to section 8.25.2)). In this case, if the Commit Action field is set to 011b, then the controller shall abort the command with a status code of Firmware Activation Requires Conventional Reset.

When modifying Boot Partitions, the host may select the Boot Partition to mark as active or replace. A Boot Partition is only able to be written when unlocked (refer to section 8.2).

The Firmware Commit command uses the Command Dword 10 field. All other command specific fields are reserved.

Figure 181: Firmware Commit – Command Dword 10

Bits	Description
31	**Boot Partition ID (BPID):** Specifies the Boot Partition that shall be used for the Commit Action, if applicable.
30:06	Reserved

Figure 181: Firmware Commit – Command Dword 10

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>05:03</td><td>Commit Action (CA): This field specifies the action that is taken (refer to section 3.11) on the image downloaded with the Firmware Image Download command or on a previously downloaded and placed image. The actions are indicated in the following table.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>Downloaded image replaces the existing image, if any, in the specified Firmware Slot. The newly placed image is not activated.</td></tr>
<tr><td>001b</td><td>Downloaded image replaces the existing image, if any, in the specified Firmware Slot. The newly placed image is activated at the next Controller Level Reset.</td></tr>
<tr><td>010b</td><td>The existing image in the specified Firmware Slot is activated at the next Controller Level Reset.</td></tr>
<tr><td>011b</td><td>Downloaded image replaces the existing image, if any, in the specified Firmware Slot and is then activated immediately. If there is not a newly downloaded image, then the existing image in the specified firmware slot is activated immediately.</td></tr>
<tr><td>100b to 101b</td><td>Reserved</td></tr>
<tr><td>110b</td><td>Downloaded image replaces the Boot Partition specified by the Boot Partition ID field.</td></tr>
<tr><td>111b</td><td>Mark the Boot Partition specified in the BPID field as active and update BPINFO.ABPID.</td></tr>
</table></td></tr>
<tr><td>02:00</td><td>Firmware Slot (FS): Specifies the firmware slot that shall be used for the Commit Action, if applicable. If the value specified is 0h, then the controller shall choose the firmware slot (i.e., slot 1 to slot 7) to use for the operation.</td></tr>
</table>

5.12.1 Command Completion

Upon completion of the Firmware Commit command, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.

For Firmware Commit commands that specify activation of a new firmware image at the next Controller Level Reset (i.e., the CA field was set to 001b or 010b) and complete with a status code value of 0h (i.e., Success Completion), a Controller Level Reset initiated by any of the methods defined in section 3.7.2 activates the specified firmware.

If the controller detects overlapping firmware/boot partition image update command sequences (refer to section 1.5.23) of more than one firmware image and/or Boot Partition or the use of more than one controller and/or Management Endpoint to update a single firmware image, then the results of that detection are reported in Dword 0 of the completion queue entry as defined in Figure 182. Refer to section 3.11 and section 8.2.2.

Figure 182: Firmware Commit – Completion Queue Entry Dword 0

Bits	Description
31:02	Reserved

Figure 182: Firmware Commit – Completion Queue Entry Dword 0

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>01:00</td><td>Multiple Update Detected (MUD): This field indicates if a controller detected overlapping firmware/boot partition image update command sequences of Boot Partitions and/or firmware images (refer to section 3.11 and section 8.2.2). If the SMUD bit in the Firmware Update field of the Identify Controller data structure is cleared to ‘0’, then this field shall be cleared to 00b.

This field is valid if the command is successful or aborted.

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>1</td><td>If set to ‘1’, then the controller detected an overlapping firmware/boot partition image update command sequence due to processing a command from a Management Endpoint. If cleared to ‘0’, then the controller did not detect an overlapping firmware/boot partition image update command sequence due to processing a command from a Management Endpoint.</td></tr>
<tr><td>0</td><td>If set to ‘1’, then the controller detected an overlapping firmware/boot partition image update command sequence due to processing a command from an Admin Submission Queue on a controller. If cleared to ‘0’, then the controller did not detect an overlapping firmware/boot partition image update command sequence due to processing a command from an Admin Submission Queue on a controller.</td></tr>
</table>
</td></tr>
</table>

Firmware Commit command specific status values (i.e., SCT field set to 1h) are shown in Figure 183.

Figure 183: Firmware Commit – Command Specific Status Values

Value	Description
06h	**Invalid Firmware Slot:** The firmware slot indicated is invalid or read only. This error is indicated if the firmware slot exceeds the number supported.
07h	**Invalid Firmware Image:** The firmware image specified for activation is: • invalid and not loaded by the controller; or • the specified firmware slot does not contain a firmware image.
0Bh	**Firmware Activation Requires Conventional Reset:** The firmware commit was successful, however, activation of the firmware image requires a Conventional Reset. If an FLR or Controller Reset occurs prior to a Conventional Reset, the controller shall continue operation with the currently executing firmware image.
10h	**Firmware Activation Requires NVM Subsystem Reset:** The firmware commit was successful, however, activation of the firmware image requires an NVM Subsystem Reset. If any other type of Controller Level Reset occurs prior to an NVM Subsystem Reset, the controller shall continue operation with the currently executing firmware image.
11h	**Firmware Activation Requires Controller Level Reset:** The firmware commit was successful; however, the image specified does not support being activated without a Controller Level Reset. The image shall be activated at the next Controller Level Reset. This status code should be returned only if the Commit Action field in the Firmware Commit command is set to 011b (i.e., activate immediately).
12h	**Firmware Activation Requires Maximum Time Violation:** The image specified if activated immediately would exceed the Maximum Time for Firmware Activation (MTFA) value reported in the Identify Controller data structure (refer to Figure 275). To activate the firmware, the Firmware Commit command needs to be re-issued and the image activated using a reset.
13h	**Firmware Activation Prohibited:** The image specified is being prohibited from activation by the controller for vendor specific reasons (e.g., controller does not support down revision firmware).
14h	**Overlapping Range**: This error is indicated if the firmware image has overlapping ranges.
1Eh	**Boot Partition Write Prohibited:** This error is indicated if a command attempts to modify a Boot Partition while locked (refer to section 8.2.3).

5.13 Firmware Image Download command

The Firmware Image Download command is used to download all or a portion of an image for a future update to the controller. The Firmware Image Download command may be submitted while other

commands on the Admin Submission Queue or I/O Submission Queues are outstanding. The Firmware Image Download command downloads a new image (in whole or in part) to the controller.

The image may be constructed of multiple pieces that are individually downloaded with separate Firmware Image Download commands. Each Firmware Image Download command includes a Dword Offset and Number of Dwords that specify a dword range. The host software should ensure that image pieces do not have dword ranges that overlap and that the NUMD field and OFST field meet the alignment and granularity requirements indicated in the FWUG field (refer to Figure 275). Firmware portions may be submitted out of order to the controller. Host software shall submit image portions in order when updating a Boot Partition. If ranges overlap, the controller may return an error of Overlapping Range.

The new firmware image is not activated as part of the Firmware Image Download command. Refer to section 3.11 for details on the firmware update process. The firmware update process does not modify the contents of Boot Partitions. Refer to section 8.2.2 for details on the Boot Partition update process.

Host software should not overlap command sequences that update Boot Partitions and/or firmware images (refer to section 3.11 and section 8.2.2).

After downloading an image, host software issues a Firmware Commit command before downloading another image. Processing of the first Firmware Image Download command after completion of a Firmware Commit command shall cause the controller to discard all remaining portion(s), if any, of downloaded images. If a reset occurs between a firmware download and completion of the Firmware Commit command, then the controller shall discard all portion(s), if any, of downloaded images.

The Firmware Image Download command uses the Data Pointer, Command Dword 10, and Command Dword 11 fields. All other command specific fields are reserved.

Figure 184: Firmware Image Download – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the location where data should be transferred from. Refer to Figure 87 for the definition of this field.

Figure 185: Firmware Image Download – Command Dword 10

Bits	Description
31:00	**Number of Dwords (NUMD):** This field specifies the number of dwords to transfer for this portion of the firmware. This is a 0's based value. If the value specified in this field does not meet the requirement indicated by the FWUG field (refer to Figure 275), the firmware update may abort with status code of Invalid Field in Command.

Figure 186: Firmware Image Download – Command Dword 11

Bits	Description
31:00	**Offset (OFST):** This field specifies the number of dwords offset from the start of the firmware image being downloaded to the controller. The offset is used to construct the complete firmware image when the firmware is downloaded in multiple pieces. The piece corresponding to the start of the firmware image shall have an Offset of 0h. If the value specified in this field does not meet the requirement indicated by the FWUG field (refer to Figure 275), the firmware update may fail with status code of Invalid Field in Command.

5.13.1 Command Completion

Upon completion of the Firmware Image Download command, the controller posts a completion queue entry to the Admin Completion Queue. Firmware Image Download command specific status values are defined in Figure 187.

Figure 187: Firmware Image Download – Command Specific Status Values

Value	Description
14h	**Overlapping Range:** This error is indicated if the firmware image has overlapping ranges. This error may indicate that the granularity or alignment of the firmware image downloaded does not conform to the Firmware Update Granularity field indicated in the Identify Controller data structure.

If the controller detects overlapping firmware/boot partition image update command sequences (refer to section 1.5.23) of more than one firmware image and/or Boot Partition or the use of more than one controller and/or Management Endpoint to update a single firmware image, then the results of that detection are reported in Dword 0 of the completion queue entry as defined in Figure 182. Refer to section 3.11 and section 8.2.2.

5.14 Format NVM command

The Format NVM command is used to low level format the NVM media. This command is used by the host to change the attributes of the NVM media (e.g., the LBA data size and/or metadata size for the NVM Command Set). A low level format may destroy all data and metadata associated with all namespaces or only the specific namespace associated with the command (refer to the Format NVM Attributes field in the Identify Controller data structure, Figure 275). After the Format NVM command successfully completes, the controller shall not return any user data that was previously contained in an affected namespace.

As part of the Format NVM command, the host requests a format operation and may request a secure erase of the contents of the NVM (refer to the SES field in Figure 189). There are two types of secure erase. The User Data Erase erases all user content present in the NVM subsystem. The Cryptographic Erase erases all user content present in the NVM subsystem by deleting the encryption key with which the user data was previously encrypted.

The scope of the format operation and the scope of the format with secure erase depend on the attributes that the controller supports for the Format NVM command and the Namespace Identifier specified in the command as described in Figure 188. The type of secure erase, if applicable, is based on the setting of the Secure Erase Settings field in Command Dword 10 as defined in Figure 189.

Figure 188: Format NVM – Operation Scope

FNA Bit [1]	NSID	Format Operation
0	FFFFFFFFh[2]	All namespaces attached to the controller. Other namespaces are not affected.
0	Any allocated value (refer to section 3.2.1.3)	Particular namespace specified. Other namespaces are not affected.
1 [3]	Any allocated value (refer to section 3.2.1.3) or FFFFFFFFh	All namespaces that exist in the NVM subsystem.

Notes:
1. For a Format NVM command with Secure Erase, this column refers to bit 1 in the FNA field in the Identify Controller data structure (refer to Figure 275) and bit 0 in the FNA field is ignored. For a Format NVM command without Secure Erase, this column refers to bit 0 in the FNA field, and bit 1 in the FNA field is ignored.
2. If bit 3 in the FNA field is set to '1', then this value is not supported.
3. If bit 3 in the FNA field is set to '1', then this value does not occur. Refer to Figure 275.

If the NVM subsystem supports multiple domains and the Format NVM command is not able to format the specified namespaces as a result of the NVM subsystem being divided (refer to section 3.2.4), then the Format NVM command shall be aborted with a status code of Asymmetric Access Inaccessible or Asymmetric Access Persistent Loss.

The Format NVM command may be aborted with a status code defined in this specification under circumstances defined by a security specification (e.g., invalid security state as specified in TCG Storage

Interface Interactions specification). If there are I/O commands being processed for a namespace, then a Format NVM command that is submitted affecting that namespace may be aborted; if aborted, then a status code of Command Sequence Error should be returned. If a Format NVM command is in progress, then an I/O command that is submitted for any namespace affected by that Format NVM command may be aborted; if aborted, then a status code of Format in Progress should be returned. Refer to section 5 for further information about restrictions on Admin Commands during Format NVM.

For a Format NVM command with the NSID field set to FFFFFFFFh that specifies secure erase:

a) if bit 1 is set to '1' in the FNA field (refer to Figure 275) and there are no namespaces in the NVM subsystem, then that Format NVM command shall complete without error; and
b) if bit 1 is cleared to '0' in the FNA field and there are no attached namespaces, then that Format NVM command shall complete without error.

For a Format NVM command with an NSID field set to FFFFFFFFh that does not specify a secure erase:

a) if bit 0 is set to '1' in the FNA field and there are no namespaces in the NVM subsystem, then that Format NVM command shall complete without error; and
b) if bit 0 is cleared to '0' in the FNA field and there are no attached namespaces, then that Format NVM command shall complete without error.

If a host does not set the LBA Format Extension Enable (LBAFEE) field to 1h in the Host Behavior Support feature (refer to section 5.27.1.18), then the 0h value of the LBAFEE field disables any I/O Command Set specific format that requires the LBAFEE field to be set to 1h (refer to the applicable I/O Command Set specification). If a Format NVM command specifies a format that is disabled (e.g., the LBAFEE field is cleared to 0h), then the controller shall abort that Format NVM command with a status code of Invalid Namespace or Format.

If the format operation scope (refer to Figure 188) for a Format NVM command includes any namespace that is write protected (refer to section 8.12), then the controller aborts that Format NVM command with a status code of Namespace is Write Protected.

If bit 3 in the FNA field is set to '1' and a Format NVM command has the NSID field set to FFFFFFFFh, then the controller shall abort the command with a status code of Invalid Field in Command.

After successful completion of a Format NVM command, the settings specified in the Format NVM command (e.g., PI, MSET, LBAF) are reported as part of the Identify Namespace data structures.

The Format NVM command uses the Command Dword 10 field. All other command specific fields are reserved.

Figure 189: Format NVM – Command Dword 10

Bits	Description
31:14	Reserved
13:12	**LBA Format Upper (LBAFU):** This field specifies the most significant 2 bits of the Format Index of the User Data Format to apply to the NVM media. This corresponds to the User Data Formats indicated in the Identify command, refer to the Identify Namespace data structure and the LBA Format data structure in the applicable I/O Command Set specification. If an unsupported User Data Format is selected, the controller shall abort the command with a status code of Invalid Format. This field is ignored if the LBA Format Extension Enable (LBAFEE) field is cleared to 0h in the Host Behavior Support feature (refer to section 5.27.1.18). NOTE: This field applies to all User Data Formats. The original name has been retained for historical continuity.

Figure 189: Format NVM – Command Dword 10

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>11:09</td><td>Secure Erase Settings (SES): This field specifies whether a secure erase should be performed as part of the format and the type of the secure erase operation. The erase applies to all user data, regardless of location (e.g., within an exposed LBA, within a cache, within deallocated logical blocks, etc.).
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>No secure erase operation requested</td></tr>
<tr><td>001b</td><td>User Data Erase: All user data shall be erased, contents of the user data after the erase is indeterminate (e.g., the user data may be zero filled, one filled, etc.). If a User Data Erase is requested and all affected user data is encrypted, then the controller is allowed to use a cryptographic erase to perform the requested User Data Erase.</td></tr>
<tr><td>010b</td><td>Cryptographic Erase: All user data shall be erased cryptographically. This is accomplished by deleting the encryption key.</td></tr>
<tr><td>011b to 111b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>08</td><td>Protection Information Location (PIL): I/O Command Set specific definition.[1]</td></tr>
<tr><td>07:05</td><td>Protection Information (PI): I/O Command Set specific definition.[1]</td></tr>
<tr><td>04</td><td>Metadata Settings (MSET): I/O Command Set specific definition.[1]</td></tr>
<tr><td>03:00</td><td>LBA Format Lower (LBAFL): This field specifies the least significant 4 bits of the Format Index of the User Data Format to apply to the NVM media. This corresponds to the User Data Formats indicated in the Identify Namespace data structure, refer to the Identify Namespace data structure and the I/O Command Set specific Format Data Structure in the applicable I/O Command Set specification. If an unsupported User Data Format is selected, the controller shall abort the command with a status code of Invalid Format.

NOTE: This field applies to all User Data Formats. The original name has been retained for historical continuity.</td></tr>
<tr><td colspan="2">NOTE:
1. I/O Command Set specific fields are described in the applicable I/O Command Set specification.</td></tr>
</table>

5.14.1 Command Completion

A completion queue entry is posted to the Admin Completion Queue when the NVM media format is complete. Format NVM command specific status values (i.e., SCT field set to 1h) are shown in Figure 190.

Figure 190: Format NVM – Command Specific Status Values

Value	Description
0Ah	**Invalid Format:** The format specified is invalid. This may be due to various conditions, including: 1. specifying an invalid User Data Format number; 2. enabling protection information when there are not sufficient metadata resources; or 3. the specified format is not available in the current configuration.
0Ch	**Command Sequence Error:** The command was aborted due to a protocol violation in a multi-command sequence.
15h	**Operation Denied:** The command was denied due to lack of access rights. Refer to the appropriate security specification.
20h	**Namespace is Write Protected:** The command is prohibited while the namespace is write protected (refer to section 8.12).
86h	**Access Denied:** Access to the namespace and/or user data is denied due to lack of access rights. Refer to the appropriate security specification (e.g., TCG Storage Interface Interactions specification).

5.15 Get Features command

The Get Features command retrieves the attributes of the Feature specified.

The Get Features command uses the Data Pointer, Command Dword 10 and Command Dword 14 fields. The use of the Command Dword 11 field is Feature specific. If not used by a Feature, then Command Dword 11 is reserved unless otherwise stated. All other command specific fields are reserved.

The mandatory, optional, and prohibited Feature Identifiers for each type of controller are defined in section 3.1.2.1.3, section 3.1.2.2.3, and section 3.1.2.3.4.

Figure 191: Get Features – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 87 for the definition of this field. If no data structure is used as part of this specified feature, then this field is ignored.

Figure 192: Get Features – Command Dword 10

Bits	Description
31:11	Reserved
10:08	**Select (SEL):** This field specifies the attribute of the value requested in the returned data: Select / Description 000b – Current 001b – Default 010b – Saved 011b – Supported Capabilities 100b to 111b – Reserved Refer to section 5.15.1 and section 4.2 for details on the data returned in each case. The controller indicates in bit 4 of the Optional NVM Command Support field of the Identify Controller data structure in Figure 275 whether this field is supported. If a Get Features command is received with the Select field set to 010b (i.e., saved) and the controller does not support the Feature Identifier being saved or does not currently have any saved values, then the controller shall operate as if the Select field is set to 001b (i.e., default).
07:00	**Feature Identifier (FID):** This field specifies the identifier of the Feature for which to provide data.

If the controller supports selection of a UUID by the Get Features command (refer to Figure 316 and section 8.25) and the controller supports selection of a UUID for the specified vendor specific Feature Identifier (refer to Figure 316), then Command Dword 14 is used to specify a UUID Index value (refer to Figure 193). If the controller does not support selection of a UUID by the Get Features command or the controller does not support selection of a UUID for the specified vendor specific Feature Identifier, then Command Dword 14 does not specify a UUID Index value.

Figure 193: Get Features – Command Dword 14

Bits	Description
31:07	Reserved
06:00	**UUID Index:** Refer to Figure 477.

Figure 194 describes the Feature Identifiers whose attributes may be retrieved using the Get Features command. The definition of the attributes returned and the associated format is specified in the section indicated.

Figure 194: Get Features – Feature Identifiers

Description	Section Defining Format of Attributes Returned
Arbitration	5.27.1.1

Figure 194: Get Features – Feature Identifiers

Description	Section Defining Format of Attributes Returned
Power Management	5.27.1.2
Temperature Threshold	5.27.1.3
Volatile Write Cache	5.27.1.4
Number of Queues	5.27.1.5
Interrupt Coalescing	5.27.1.6
Interrupt Vector Configuration	5.27.1.7
Asynchronous Event Configuration	5.27.1.8
Autonomous Power State Transition	5.27.1.9
Host Memory Buffer	5.27.1.10
Timestamp	5.27.1.11
Keep Alive Timer	5.27.1.12
Host Controlled Thermal Management	5.27.1.13
Non-Operational Power State Config	5.27.1.14
Read Recovery Level Config	5.27.1.15
Predictable Latency Mode Config	5.27.1.16
Predictable Latency Mode Window	5.27.1.17
Host Behavior Support	5.27.1.18
Sanitize Config	5.27.1.19
Endurance Group Event Configuration	5.27.1.20
I/O Command Set Profile	5.27.1.21
Software Progress Marker	5.27.1.24
Host Identifier	5.27.1.25
Reservation Notification Mask	5.27.1.26
Reservation Persistence	5.27.1.27
Namespace Write Protection Config	5.27.1.28
Enhanced Controller Metadata	5.27.1.23.1
Controller Metadata	5.27.1.23.2
Namespace Metadata	5.27.1.23.3
Spinup Control	5.27.1.22
I/O Command Set specific features	I/O Command Set specification

5.15.1 Select field

A Select field cleared to 000b (i.e., current) returns the current operating attribute value for the Feature Identifier specified.

A Select field set to 001b (i.e., default) returns the default attribute value for the Feature Identifier specified.

A Select field set to 010b (i.e., saved) returns the last saved attribute value for the Feature Identifier specified (i.e., the last Set Features command completed without error, with the Save bit set to ‘1’ for the Feature Identifier specified).

A Select field set to 011b (i.e., supported capabilities) returns the capabilities supported for this Feature Identifier. The capabilities supported are returned in Dword 0 of the completion queue entry of the Get Features command (refer to Figure 195).

5.15.2 Command Completion

Upon completion of the Get Features command, the controller posts a completion queue entry to the Admin Completion Queue. If the Select field is not set to 011b, then Dword 0 of the completion queue entry may contain feature-dependent information (refer to section 5.27.1).

If the Select field is set to 011b, then Figure 195 describes the contents of Dword 0 of the completion queue entry.

Figure 195: Completion Queue Entry Dword 0 when Select is set to 11b

Bits	Description
31:3	Reserved
2	**Changeable**: If set to '1', then the feature values are changeable. If cleared to '0', then the feature values are not changeable.
1	**NS Specific**: If set to '1', then the Feature Identifier is namespace specific and settings are applied to individual namespaces. If cleared to '0', then the Feature Identifier is not namespace specific and its settings apply to the entire controller.
0	**Saveable**: If set to '1', then the feature values are saveable. If cleared to '0', then the feature values are not saveable.

5.16 Get Log Page command

The Get Log Page command returns a data buffer containing the log page requested. The Get Log Page command may be impacted by the ANA state (refer to section 8.1.4).

The Get Log Page command uses the Data Pointer, Command Dword 10, Command Dword 11, Command Dword 12, Command Dword 13, and Command Dword 14 fields. All other command specific fields are reserved.

There are mandatory and optional Log Identifiers defined in section 3.1.2.1.2, section 3.1.2.2.2, and section 3.1.2.3.3. If a Get Log Page command is processed that specifies a Log Identifier that is not supported, then the controller should abort the command with a status code of Invalid Log Page with the exception defined in Figure 268.

The controller indicates support for the Log Page Offset and extended Number of Dwords (32 bits rather than 12 bits) in the Log Page Attributes field of the Identify Controller data structure. If extended data is not supported, then bits 27:16 of the Number of Dwords Lower field specify the Number of Dwords to transfer.

If the Log Page Offset is supported, then

- a byte offset shall be supported (i.e., Offset Type field cleared to '0') for all log pages; and
- for each log page that has the IOS bit set to '1' for the specified LID in the LID Supported and Effects Data Structure log page (refer to Figure 204) an index offset shall be supported (i.e., Offset Type field set to '1').

If the IOS bit is cleared to '0' for the specified LID in the LID Supported and Effects Data Structure log page and a Get Log Page command specifies Offset Type field set to '1', then that command shall be aborted with a status code of Invalid Field in Command.

Figure 196: Get Log Page – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 87 for the definition of this field.

Figure 197: Get Log Page – Command Dword 10

Bits	Description
31:16	**Number of Dwords Lower (NUMDL):** This field specifies the least significant 16 bits of the number of dwords to return unless otherwise specified. If host software specifies a size larger than the log page requested, the controller returns the complete log page with undefined results for dwords beyond the end of the log page. The combined NUMDL and NUMDU fields form a 0's based value.

Figure 197: Get Log Page – Command Dword 10

Bits	Description
15	**Retain Asynchronous Event (RAE):** This bit specifies whether to retain or clear an Asynchronous Event. If this bit is cleared to '0', the corresponding Asynchronous Event is cleared by the controller upon successful command completions. If this bit is set to '1', the corresponding Asynchronous Event is retained (i.e., not cleared) by the controller upon command completion. If the command does not complete successfully, the Asynchronous Event shall be retained by the controller. Host software should clear this bit to '0' for log pages that are not used with Asynchronous Events. Refer to section 5.2.
14:08	**Log Specific Parameter (LSP):** If not defined for the log specified by the Log Page Identifier field, this field is reserved**.**
07:00	**Log Page Identifier (LID):** This field specifies the identifier of the log page to retrieve.

Figure 198: Get Log Page – Command Dword 11

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:16</td><td>Log Specific Identifier: This field specifies an identifier that is required for a particular log page. The log pages that require a log specific identifier are indicated in the table below.
<table>
<tr><th>Log Page</th><th>Identifier</th></tr>
<tr><td>Endurance Group Information</td><td rowspan="2">Endurance Group Identifier (refer to section 3.2.3)</td></tr>
<tr><td>Rotational Media Information</td></tr>
<tr><td>Predictable Latency Per NVM Set</td><td>NVM Set Identifier (refer to section 3.2.2)</td></tr>
<tr><td>Media Unit Status</td><td>Domain Identifier (refer to section 3.2.4) [1]</td></tr>
<tr><td>Supported Capacity Configuration List</td><td>Domain Identifier (refer to section 3.2.4) [1]</td></tr>
</table></td></tr>
<tr><td>15:00</td><td>Number of Dwords (NUMDU): This field specifies the most significant 16 bits of the number of dwords to return unless otherwise specified.</td></tr>
<tr><td colspan="2">Notes:
1. If the NVM subsystem does not support multiple domains, then this field is reserved. If this field specifies a non-zero Domain Identifier that is not reported in the Domain List (refer to section 5.17.2.17), then the controller shall abort the command with Invalid Field in Command.</td></tr>
</table>

Figure 199: Get Log Page – Command Dword 12

Bits	Description
31:00	**Log Page Offset Lower (LPOL):** The log page offset specifies the location within a log page to start returning data from unless otherwise specified. If the OT bit is cleared to '0', then: a) This field specifies the least significant 32 bits of the log page offset. The offset shall be dword aligned, indicated by bits 1:0 being cleared to 00b; b) The controller is not required to check that bits 1:0 are cleared to 00b. The controller may report an error of Invalid Field in Command if bits 1:0 are not cleared to 00b. If the controller does not report an error of Invalid Field in Command, then the controller shall operate as if bits 1:0 are cleared to 00b; and c) If the host specifies an offset (i.e., LPOL and LPOU) that is greater than the size of the log page requested (e.g., a log page containing 100 bytes is requested starting at offset 200), then the controller shall abort the command with a status of Invalid Field in Command. If the OT bit is set to '1', then: a) This field specifies the least significant 32 bits of the index into the list of data structures in the log page; b) If the host specifies an index (i.e., LPOL and LPOU) that is greater than the number of entries in the list of data structures in the log page requested (e.g., a log page containing 100 data structures is requested starting at index 200), then the controller shall abort the command with a status code of Invalid Field in Command; c) If the IOS bit for the specified LID in the LID Supported and Effects Data Structure log page is cleared to '0', then the controller shall abort the command with a status code of Invalid Field in Command; and d) Each log page that supports the use of an index offset value defines the contents of an entry for the purposes of indexing into that log page.

Figure 200: Get Log Page – Command Dword 13

Bits	Description
31:00	**Log Page Offset Upper (LPOU):** This field specifies the most significant 32 bits of either the log page offset or the index into the list of data structures unless otherwise specified. Refer to the Log Page Offset Lower definition.

If the controller supports selection of a UUID by the Get Log Page command (refer to Figure 202 and section 8.25), then Command Dword 14 is used to specify a UUID Index value (refer to Figure 201).

Figure 201: Get Log Page – Command Dword 14

Bits	Description
31:24	**Command Set Identifier (CSI)**: Refer to Figure 274.
23	**Offset Type (OT):** If set to '1' then the Log Page Offset Lower field and the Log Page Offset Upper field specify the index into the list of data structures in the log page to be returned. If cleared to '0', then the Log Page Offset Lower field and the Log Page Offset Upper field specify the byte offset into the log page to be returned.
22:07	Reserved
06:00	**UUID Index:** Refer to Figure 477.

5.16.1 Log Specific Information

Figure 202 defines the log pages that may be retrieved with the Get Log Page command and the scope of the information that is returned in those log pages. Refer to section 3.1.2.1.2, section 3.1.2.2.2, and 3.1.2.3.3 for mandatory, optional, and prohibited log pages for the various controller types.

Log pages that indicate a scope of NVM subsystem return information that is global to the NVM subsystem. Log pages that indicate a scope of Domain return information that is global to the Domain. Log pages that indicate a scope of controller return information that is specific to the controller that is processing the command. Log pages that indicate a scope of Namespace return information that is specific to the specified namespace. For log pages that indicate multiple scopes, support for multiple domains or the namespace identifier that is specified determines which information is returned. The definition of any individual field within a log page may indicate a different scope that is specific to that individual field.

For log pages with a scope of NVM subsystem or controller (as shown in Figure 202), the controller should abort commands that specify namespace identifiers other than 0h or FFFFFFFFh with status code Invalid Field in Command. Otherwise the rules for namespace identifier usage in Figure 87 apply.

Figure 202: Get Log Page – Log Page Identifiers

Log Identifier	Scope	Log Page Name	Reference Section
00h	Controller	Supported Log Pages	5.16.1.1
01h	Controller	Error Information	5.16.1.2
02h	Controller [1] Namespace [2]	SMART / Health Information	5.16.1.3
03h	Domain / NVM subsystem [6]	Firmware Slot Information	5.16.1.4
04h	Controller	Changed Namespace List	5.16.1.5
05h	Controller	Commands Supported and Effects	5.16.1.6
06h	Controller [3] Domain / NVM subsystem [4, 6]	Device Self-test [5]	5.16.1.7
07h	Vendor Specific	Telemetry Host-Initiated [5]	5.16.1.8
08h	Vendor Specific	Telemetry Controller-Initiated [5]	5.16.1.9
09h	Domain / NVM subsystem [6]	Endurance Group Information	5.16.1.10
0Ah	Domain / NVM subsystem [6]	Predictable Latency Per NVM Set	5.16.1.11
0Bh	Domain / NVM subsystem [6]	Predictable Latency Event Aggregate	5.16.1.12
0Ch	Controller	Asymmetric Namespace Access	5.16.1.13
0Dh	NVM subsystem	Persistent Event Log [5]	5.16.1.14
0Eh	Refer to the NVM Command Set		
0Fh	Domain / NVM subsystem [6]	Endurance Group Event Aggregate	5.16.1.15
10h	Domain / NVM subsystem [5, 6]	Media Unit Status	5.16.1.16
11h	Domain / NVM subsystem [6]	Supported Capacity Configuration List	5.16.1.17
12h	Controller	Feature Identifiers Supported and Effects	5.16.1.18
13h	Controller	NVMe-MI Commands Supported and Effects	5.16.1.19
14h	NVM subsystem	Command and Feature Lockdown [5]	5.16.1.20
15h	Controller	Boot Partition	5.16.1.21
16h	Endurance Group	Rotational Media Information	5.16.1.22
17h to 6Fh	Reserved		
70h		Discovery	5.16.1.23
71h to 7Fh	Reserved		
80h	Controller	Reservation Notification	5.16.1.24
81h	NVM subsystem	Sanitize Status	5.16.1.25
82h to BEh	I/O Command Set Specific		
BFh	Refer to the Zoned Namespace Command Set		
C0h to FFh	Vendor specific [5]		

Key:
Namespace = The log page contains information about a specific namespace.
Endurance Group = The log page contains information about a specific Endurance Group.
Controller = The log page contains information about the controller that is processing the command.
Domain = The log page contains information about the Domain.
NVM subsystem = The log page contains information about the NVM subsystem.
Vendor Specific = The log page contains information that is vendor specific.

Notes:
1. For namespace identifiers of 0h or FFFFFFFFh.
2. For namespace identifiers other than 0h or FFFFFFFFh.
3. Bit 0 is cleared to ‘0’ in the DSTO field in the Identify Controller data structure (refer to Figure 275).
4. Bit 0 is set to ‘1’ in the DSTO field in the Identify Controller data structure.

5. Selection of a UUID may be supported. Refer to section 8.25.
6. For NVM subsystems that support multiple domains (refer to the MDS bit in the Identify Controller data structure, Figure 275), Domain scope information is returned.

5.16.1.1 Supported Log Pages (Log Identifier 00h)

An NVM subsystem may support several interfaces for submitting a Get Log Page command such as an Admin Submission Queue, PCIe VDM Management Endpoint, or SMBus/I2C Management Endpoint (refer to the NVM Express Management Interface Specification for details on Management Endpoints) and may have zero or more instances of each of those interfaces. The log pages supported on each instance of each interface may be different. This log page is used to describe the log pages that are supported on the interface to which the Get Log Page command was submitted and attributes specific to each log page. The log page is defined in Figure 203. The attributes of each log page are described in a LID Supported and Effects data structure defined in Figure 204.

If the UUID Selection Supported bit is set to '1' for the Get Log Page command in the Commands Supported and Effects log page (refer to section 5.16.1.6), then the log page data reflects the log pages that are supported based on the value of the UUID Index field (refer to section 8.25).

The log pages that the controller supports are dependent on the I/O Command Set that is based on:

- the I/O Command Set selected in CC.CSS, if CC.CSS is not set to 110b; and
- the Command Set Identifier (CSI) field in CDW 14, if CC.CSS is set to 110b.

Figure 203: Supported Log Pages Log Page

Bytes	Description
3:0	**Log Page Identifier Supported 0:** Contains the LID Supported and Effects data structure (refer to Figure 204.) for the LID 0h.
7:4	**Log Page Identifier Supported 1:** Contains the LID Supported and Effects data structure (refer to Figure 204.) for the LID 1h.
…	**…**
1019:1016	**Log Page Identifier Supported 254:** Contains the LID Supported and Effects data structure (refer to Figure 204.) for the LID FEh.
1023:1020	**Log Page Identifier Supported 255:** Contains the LID Supported and Effects data structure (refer to Figure 204.) for the LID FFh.

Figure 204: LID Supported and Effects Data Structure

Bits	Description
31:16	**LID Specific Parameter (LIDSP):** This field is specific to the log page identifier as defined in Figure 205.
15:2	Reserved
1	**Index Offset Supported (IOS):** If this bit is set to '1', then the controller supports an index offset for this LID in a Get Log Page command (i.e., the OT bit in the Get Log Page command is allowed to be set to '1'). If this bit is cleared to '0', then the controller does not support an index offset for this LID in a Get Log Page command (i.e., the OT bit in the Get Log Page command is only allowed to be cleared to '0').
0	**LID Supported (LSUPP):** If this bit is set to '1', then the controller supports this LID for a Get Log Page command. If this bit is cleared to '0', then the controller does not support this LID for a Get Log Page command. Refer to section 3.1.2 for the LID support requirements for each controller type.

Figure 205: LID Supported and Effects Data Structure – LID Specific Parameter Field

Log Page Identifier	LID Specific Parameter Field
0 to Ch	Reserved

Figure 205: LID Supported and Effects Data Structure – LID Specific Parameter Field

<table>
<tr><th>Log Page Identifier</th><th>LID Specific Parameter Field</th></tr>
<tr><td>0Dh</td><td>The LID Specific Parameter field for log page identifier 0Dh (Persistent Event log page as described in section 5.16.1.14) is defined as follows:
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>15:1</td><td>Reserved</td></tr>
<tr><td>0</td><td>Establish Context and Read 512 Bytes of Header Supported: If this bit is cleared to ‘0’, then the controller does not support the Establish Context and Read 512 Bytes of Header action (refer to Figure 223).

If this bit is set to ‘1’, then the controller supports:
• the Establish Context and Read 512 Bytes of Header action; and
• the Generation Number field in the Persistent Event log page.

Implementations compliant with NVM Express Base Specification, Revision 2.0 and later shall set this bit to ‘1’.</td></tr>
</table></td></tr>
<tr><td>0Eh to BFh</td><td>Reserved</td></tr>
<tr><td>C0h to FFh</td><td>Vendor specific</td></tr>
</table>

5.16.1.2 Error Information (Log Identifier 01h)

This log page is used to describe extended error information for a command that completed with error or report an error that is not specific to a particular command. Extended error information is provided when the More (M) bit is set to ‘1’ in the Status field for the completion queue entry associated with the command that completed with error or as part of an asynchronous event with an Error status type. This log page is global to the controller.

This error log may return the last *n* errors. If host software specifies a data transfer of the size of *n* error logs, then the error logs for the most recent *n* errors are returned. The ordering of the entries is based on the time when the error occurred, with the most recent error being returned as the first log entry.

Each entry in the log page returned is defined in Figure 206. The log page is a set of 64-byte entries; the maximum number of entries supported is indicated in the ELPE field in the Identify Controller data structure (refer to Figure 275). If the log page is full when a new entry is generated, the controller should insert the new entry into the log and discard the oldest entry.

The controller should clear this log page by removing all entries on power cycle and Controller Level Reset.

Figure 206: Error Information Log Entry Data Structure

Bytes	Description
07:00	**Error Count:** This is a 64-bit incrementing error count, indicating a unique identifier for this error. The error count starts at 1h, is incremented for each unique error log entry, and is retained across power off conditions. A value of 0h indicates an invalid entry; this value is used when there are lost entries or when there are fewer errors than the maximum number of entries the controller supports. If the value of this field is FFFFFFFF_FFFFFFFFh, then the field shall be set to 1h when incremented (i.e., rolls over to 1h). Prior to NVMe 1.4, processing of incrementing beyond FFFFFFFFh is unspecified.
09:08	**Submission Queue ID:** This field indicates the Submission Queue Identifier of the command that the error information is associated with. If the error is not specific to a particular command, then this field shall be set to FFFFh.
11:10	**Command ID:** This field indicates the Command Identifier of the command that the error is associated with. If the error is not specific to a particular command, then this field shall be set to FFFFh.

Figure 206: Error Information Log Entry Data Structure

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>13:12</td><td>
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>15:1</td><td>Status: This field indicates the Status field for the command that completed. If the error is not specific to a particular command, then this field reports the most applicable status value.</td></tr>
<tr><td>0</td><td>Phase Tag: This field may indicate the Phase Tag posted for the command.</td></tr>
</table>
</td></tr>
<tr><td>15:14</td><td>Parameter Error Location: This field indicates the byte and bit of the command parameter that the error is associated with, if applicable. If the parameter spans multiple bytes or bits, then the location indicates the least-significant byte and bit of the parameter.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>15:11</td><td>Reserved</td></tr>
<tr><td>10:08</td><td>Bit Location: The offset in the byte specified by the Byte Location field to the bit in that byte that contained the error.</td></tr>
<tr><td>07:00</td><td>Byte Location: The offset in the submission queue entry to the byte in the command that contained the error. Valid values are based on the SQES field (refer to Figure 275) (e.g., a value of 6 in SQES indicates that the valid values for this field are 0 to 63).</td></tr>
</table>
If the error is not specific to a particular command, then this field shall be set to FFFFh.</td></tr>
<tr><td>23:16</td><td>LBA: This field indicates I/O Command Set specific data about the error condition, if applicable. The description is described in the applicable I/O Command Set specification.

NOTE: The original field name has been retained for historical continuity.</td></tr>
<tr><td>27:24</td><td>Namespace: This field indicates the NSID of the namespace that the error is associated with, if applicable.</td></tr>
<tr><td>28</td><td>Vendor Specific Information Available: If there is additional vendor specific error information available, this field provides the log page identifier associated with that page. A value of 0h indicates that no additional information is available. Valid values are in the range of 80h to FFh.</td></tr>
<tr><td>29</td><td>Transport Type (TRTYPE): This field indicates the Transport Type of the transport associated with the error. The values in this field are the same as the TRTYPE values in the Discovery Log Page Entry (refer to section 5.16.1.21). If the error is not transport related, this field shall be cleared to 0h. If the error is transport related, this field shall be set to the type of the transport as defined in the TRTYPE field within Figure 264.</td></tr>
<tr><td>31:30</td><td>Reserved</td></tr>
<tr><td>39:32</td><td>Command Specific Information: This field contains command specific information. If used, the command definition specifies the information returned.</td></tr>
<tr><td>41:40</td><td>Transport Type Specific Information: This field indicates additional transport type specific error information. If multiple errors exist, then this field indicates additional information about the first error. This field is transport type dependent (refer to TRTYPE) as follows:
<table>
<tr><th>Transport Type</th><th>Description</th></tr>
<tr><td>All other values</td><td>Reserved</td></tr>
<tr><td>3h</td><td>This field indicates, the offset, in bytes, from the start of the Transport Header to the start of the field that is in error. If multiple errors exist, then this field indicates the lowest offset that is in error.</td></tr>
</table>
</td></tr>
<tr><td>63:42</td><td>Reserved</td></tr>
</table>

5.16.1.3 SMART / Health Information (Log Identifier 02h)

This log page is used to provide SMART and general health information. The information provided is over the life of the controller and is retained across power cycles unless otherwise specified. To request the controller log page, the namespace identifier specified is FFFFFFFFh or 0h. For compatibility with implementations compliant with NVM Express Base Specification, Revision 1.4 and earlier, hosts should use a namespace identifier of FFFFFFFFh to request the controller log page. The controller may also

support requesting the log page on a per namespace basis, as indicated by bit 0 of the LPA field in the Identify Controller data structure in Figure 275.

If the log page is not supported on a per namespace basis, specifying a namespace identifier other than 0h or FFFFFFFFh should abort the command with a status code of Invalid Field in Command. If the controller does not abort the command, then the controller returns the controller log page. There is no namespace specific information defined in the SMART / Health Information log page in this revision of the specification, thus the controller log page and namespaces specific log page contain identical information.

Critical warnings regarding the health of the NVM subsystem may be indicated via an asynchronous event notification to the host. The warnings that results in an asynchronous event notification to the host are configured using the Set Features command; refer to section 5.27.1.8.

Performance may be calculated using parameters returned as part of the SMART / Health Information log. Specifically, the number of Read or Write commands, the amount of data read or written, and the amount of controller busy time enables both I/Os per second and bandwidth to be calculated.

The log page returned is defined in Figure 207.

Figure 207: SMART / Health Information Log Page

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>00</td><td>Critical Warning: This field indicates critical warnings for the state of the controller. Each bit corresponds to a critical warning type; multiple bits may be set to ‘1’. If a bit is cleared to ‘0’, then that critical warning does not apply. Critical warnings may result in an asynchronous event notification to the host. Bits in this field represent the state at the time the Get Log Page command is processed and may not reflect the state at the time a related asynchronous event notification, if any, occurs or occurred.
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>7:6</td><td>Reserved</td></tr>
<tr><td>5</td><td>If set to ‘1’, then the Persistent Memory Region has become read-only or unreliable (refer to section 8.14).</td></tr>
<tr><td>4</td><td>If set to ‘1’, then the volatile memory backup device has failed. This field is only valid if the controller has a volatile memory backup solution.</td></tr>
<tr><td>3</td><td>If set to ‘1’, then all of the media has been placed in read only mode. The controller shall not set this bit to '1' if the read-only condition on the media is a result of a change in the write protection state of a namespace (refer to section 8.12.1).</td></tr>
<tr><td>2</td><td>If set to ‘1’, then the NVM subsystem reliability has been degraded due to significant media related errors or any internal error that degrades NVM subsystem reliability.</td></tr>
<tr><td>1</td><td>If set to ‘1’, then a temperature is:
a) greater than or equal to an over temperature threshold; or
b) less than or equal to an under temperature threshold,
(refer to section 5.27.1.3).</td></tr>
<tr><td>0</td><td>If set to ‘1’, then the available spare capacity has fallen below the threshold.</td></tr>
</table></td></tr>
<tr><td>02:01</td><td>Composite Temperature: 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 manner in which this value is computed is implementation specific and may not represent the actual temperature of any physical point in the NVM subsystem. The value of this field may be used to trigger an asynchronous event (refer to section 5.27.1.3).
Warning and critical overheating composite temperature threshold values are reported by the WCTEMP and CCTEMP fields in the Identify Controller data structure in Figure 275.</td></tr>
<tr><td>03</td><td>Available Spare: Contains a normalized percentage (0% to 100%) of the remaining spare capacity available.</td></tr>
<tr><td>04</td><td>Available Spare Threshold: When the Available Spare falls below the threshold indicated in this field, an asynchronous event completion may occur. The value is indicated as a normalized percentage (0% to 100%). The values 101 to 255 are reserved.</td></tr>
</table>

Figure 207: SMART / Health Information Log Page

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>05</td><td>Percentage Used: 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. 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 shall be updated once per power-on hour (when the controller is not in a sleep state).

Refer to the JEDEC JESD218A standard for SSD device life and endurance measurement techniques.</td></tr>
<tr><td>06</td><td>Endurance Group Critical Warning Summary: This field indicates critical warnings for the state of Endurance Groups. Each bit corresponds to a critical warning type, multiple bits may be set to ‘1’. If a bit is cleared to ‘0’, then that critical warning does not apply to any Endurance Group. Critical warnings may result in an asynchronous event notification to the host. Bits in this field represent the current associated state and are not persistent.
If a bit is set to ‘1’ in one or more Endurance Groups, then the corresponding bit shall be set to ‘1’ in this field.

<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>7:4</td><td>Reserved</td></tr>
<tr><td>3</td><td>If set to ‘1’, then the namespaces in one or more Endurance Groups have been placed in read only mode not as a result of a change in the write protection state of a namespace (refer to section 8.12.1).</td></tr>
<tr><td>2</td><td>If set to ‘1’, then the reliability of one or more Endurance Groups has been degraded due to significant media related errors or any internal error that degrades NVM subsystem reliability.</td></tr>
<tr><td>1</td><td>Reserved</td></tr>
<tr><td>0</td><td>If set to ‘1’, then the available spare capacity of one or more Endurance Groups has fallen below the threshold.</td></tr>
</table></td></tr>
<tr><td>31:07</td><td>Reserved</td></tr>
<tr><td>47:32</td><td>Data Units Read: Contains the number of 512 byte data units the host has read from the controller as part of processing a SMART Data Units Read Command; this value does not include metadata. This value is reported in thousands (i.e., a value of 1 corresponds to 1,000 units of 512 bytes read) and is rounded up (e.g., one indicates that the number of 512 byte data units read is from 1 to 1,000, three indicates that the number of 512 byte data units read is from 2,001 to 3,000).

Refer to the specific I/O Command Set specification for the list of SMART Data Units Read Commands that affect this field.

A value of 0h in this field indicates that the number of SMART Data Units Read is not reported.</td></tr>
<tr><td>63:48</td><td>Data Units Written: Contains the number of 512 byte data units the host has written to the controller as part of processing a User Data Out Command; this value does not include metadata. This value is reported in thousands (i.e., a value of 1 corresponds to 1,000 units of 512 bytes written) and is rounded up (e.g., one indicates that the number of 512 byte data units written is from 1 to 1,000, three indicates that the number of 512 byte data units written is from 2,001 to 3,000).

Refer to the specific I/O Command Set specification for the list of User Data Out Commands that affect this field.

A value of 0h in this field indicates that the number of Data Units Written is not reported.</td></tr>
<tr><td>79:64</td><td>Host Read Commands: Contains the number of SMART Host Read Commands completed by the controller.

Refer to the specific I/O Command Set specification for the list of SMART Host Read Commands that affect this field.</td></tr>
</table>

Figure 207: SMART / Health Information Log Page

Bytes	Description
95:80	**Host Write Commands:** Contains the number of User Data Out Commands completed by the controller. Refer to the specific I/O Command Set specification for the list of User Data Out Commands that affect this field.
111:96	**Controller Busy Time:** Contains the amount of time the controller is busy with I/O commands. The controller is busy when there is a command outstanding to an I/O Queue (specifically, a command was issued via an I/O Submission Queue Tail doorbell write and the corresponding completion queue entry has not been posted yet to the associated I/O Completion Queue). This value is reported in minutes.
127:112	**Power Cycles:** Contains the number of power cycles.
143:128	**Power On Hours:** Contains the number of power-on hours. This may not include time that the controller was powered and in a non-operational power state.
159:144	**Unsafe Shutdowns:** Contains the number of unsafe shutdowns. This count is incremented when the controller does not report it is safe to power down prior to loss of main power. If CAP.CPS is cleared to 00b or set to 01b, it is safe to power down the controller when a controller shutdown processing is complete (i.e., CSTS.ST is cleared to '0' and CSTS.SHST is set to 10b). If CAP.CPS is set to 10b, it is safe to power down the domain when NVM Subsystem Shutdown processing is complete (i.e., CSTS.ST is set to '1' and CSTS.SHST is set to 10b). If CAP.CPS is set to 11b, it is safe to power down the NVM subsystem when NVM Subsystem Shutdown processing is complete (i.e., CSTS.ST is set to '1' and CSTS.SHST is set to 10b).
175:160	**Media and Data Integrity Errors:** Contains the number of occurrences where the controller detected an unrecovered data integrity error. Errors such as uncorrectable ECC, CRC checksum failure, or LBA tag mismatch are included in this field. Errors introduced as a result of a Write Uncorrectable command (refer to the NVM Command Set Specification) may or may not be included in this field.
191:176	**Number of Error Information Log Entries:** Contains the number of Error Information log Entries over the life of the controller.
195:192	**Warning Composite Temperature Time:** Contains the amount of time in minutes that the controller is operational and the Composite Temperature is greater than or equal to the Warning Composite Temperature Threshold (WCTEMP) field and less than the Critical Composite Temperature Threshold (CCTEMP) field in the Identify Controller data structure in Figure 275. If the value of the WCTEMP or CCTEMP field is 0h, then this field is always cleared to 0h regardless of the Composite Temperature value.
199:196	**Critical Composite Temperature Time:** Contains the amount of time in minutes that the controller is operational and the Composite Temperature is greater than or equal to the Critical Composite Temperature Threshold (CCTEMP) field in the Identify Controller data structure in Figure 275. If the value of the CCTEMP field is 0h, then this field is always cleared to 0h regardless of the Composite Temperature value.
201:200	**Temperature Sensor 1:** Contains the current temperature reported by temperature sensor 1. This field is defined by Figure 208.
203:202	**Temperature Sensor 2:** Contains the current temperature reported by temperature sensor 2. This field is defined by Figure 208.
205:204	**Temperature Sensor 3:** Contains the current temperature reported by temperature sensor 3. This field is defined by Figure 208.
207:206	**Temperature Sensor 4:** Contains the current temperature reported by temperature sensor 4. This field is defined by Figure 208.
209:208	**Temperature Sensor 5:** Contains the current temperature reported by temperature sensor 5. This field is defined by Figure 208.
211:210	**Temperature Sensor 6:** Contains the current temperature reported by temperature sensor 6. This field is defined by Figure 208.

Figure 207: SMART / Health Information Log Page

Bytes	Description
213:212	**Temperature Sensor 7:** Contains the current temperature reported by temperature sensor 7. This field is defined by Figure 208.
215:214	**Temperature Sensor 8:** Contains the current temperature reported by temperature sensor 8. This field is defined by Figure 208.
219:216	**Thermal Management Temperature 1 Transition Count:** Contains the number of times the controller transitioned to lower power active power states or performed vendor specific thermal management actions while minimizing the impact on performance in order to attempt to reduce the Composite Temperature because of the host controlled thermal management feature (refer to section 8.15.5) (i.e., the Composite Temperature rose above the Thermal Management Temperature 1). This counter shall not wrap once the value FFFFFFFFh is reached. A value of 0h, indicates that this transition has never occurred or this field is not implemented.
223:220	**Thermal Management Temperature 2 Transition Count:** Contains the number of times the controller transitioned to lower power active power states or performed vendor specific thermal management actions regardless of the impact on performance (e.g., heavy throttling) in order to attempt to reduce the Composite Temperature because of the host controlled thermal management feature (refer to section 8.15.5) (i.e., the Composite Temperature rose above the Thermal Management Temperature 2). This counter shall not wrap once the value FFFFFFFFh is reached. A value of 0h, indicates that this transition has never occurred or this field is not implemented.
227:224	**Total Time For Thermal Management Temperature 1:** Contains the number of seconds that the controller had transitioned to lower power active power states or performed vendor specific thermal management actions while minimizing the impact on performance in order to attempt to reduce the Composite Temperature because of the host controlled thermal management feature (refer to section 8.15.5). This counter shall not wrap once the value FFFFFFFFh is reached. A value of 0h, indicates that this transition has never occurred or this field is not implemented.
231:228	**Total Time For Thermal Management Temperature 2:** Contains the number of seconds that the controller had transitioned to lower power active power states or performed vendor specific thermal management actions regardless of the impact on performance (e.g., heavy throttling) in order to attempt to reduce the Composite Temperature because of the host controlled thermal management feature (refer to section 8.15.5). This counter shall not wrap once the value FFFFFFFFh is reached. A value of 0h, indicates that this transition has never occurred or this field is not implemented.
511:232	Reserved

Figure 208: Temperature Sensor Data Structure

Bits	Description
15:00	**Temperature Sensor Temperature (TST):** Contains the current temperature in Kelvins reported by the temperature sensor. The physical point in the NVM subsystem whose temperature is reported by the temperature sensor and the temperature accuracy is implementation specific. An implementation that does not implement the temperature sensor reports a value of 0h. The temperature reported by a temperature sensor may be used to trigger an asynchronous event (refer to section 5.27.1.3).

5.16.1.4 Firmware Slot Information (Log Identifier 03h)

This log page is used to describe the firmware revision stored in each firmware slot supported. The firmware revision is indicated as an ASCII string. The log page also indicates the active slot number. The log page returned is defined in Figure 209.

Figure 209: Firmware Slot Information Log Page

Bytes	Description
00	**Active Firmware Info (AFI):** Specifies information about the active firmware revision. Bit 7 is reserved. Bits 6:4 indicates the firmware slot that is going to be activated at the next Controller Level Reset. If this field is 0h, then the controller does not indicate the firmware slot that is going to be activated at the next Controller Level Reset. Bit 3 is reserved. Bits 2:0 indicates the firmware slot from which the actively running firmware revision was loaded.
07:01	Reserved
15:08	**Firmware Revision for Slot 1 (FRS1):** Contains the revision of the firmware downloaded to firmware slot 1. If no valid firmware revision is present or if this slot is unsupported, this field shall be cleared to 0h.
23:16	**Firmware Revision for Slot 2 (FRS2):** Contains the revision of the firmware downloaded to firmware slot 2. If no valid firmware revision is present or if this slot is unsupported, this field shall be cleared to 0h.
31:24	**Firmware Revision for Slot 3 (FRS3):** Contains the revision of the firmware downloaded to firmware slot 3. If no valid firmware revision is present or if this slot is unsupported, this field shall be cleared to 0h.
39:32	**Firmware Revision for Slot 4 (FRS4):** Contains the revision of the firmware downloaded to firmware slot 4. If no valid firmware revision is present or if this slot is unsupported, this field shall be cleared to 0h.
47:40	**Firmware Revision for Slot 5 (FRS5):** Contains the revision of the firmware downloaded to firmware slot 5. If no valid firmware revision is present or if this slot is unsupported, this field shall be cleared to 0h.
55:48	**Firmware Revision for Slot 6 (FRS6):** Contains the revision of the firmware downloaded to firmware slot 6. If no valid firmware revision is present or if this slot is unsupported, this field shall be cleared to 0h.
63:56	**Firmware Revision for Slot 7 (FRS7):** Contains the revision of the firmware downloaded to firmware slot 7. If no valid firmware revision is present or if this slot is unsupported, this field shall be cleared to 0h.
511:64	Reserved

5.16.1.5 Changed Namespace List (Log Identifier 04h)

This log page is used to describe namespaces attached to the controller that have:

a) changed information in their Identify Namespace data structures (refer to in Figure 147) since the last time the log page was read;
b) changed information in their I/O Command Set Independent Identify Namespace data structure since the last time the log page was read;
c) been added; and
d) been deleted.

The log page contains a Namespace List with up to 1,024 entries. If more than 1,024 namespaces have changed attributes since the last time the log page was read, the first entry in the log page shall be set to FFFFFFFFh and the remainder of the list shall be zero filled.

5.16.1.6 Commands Supported and Effects (Log Identifier 05h)

This log page is used to describe the commands that the controller supports and the effects of those commands on the state of the NVM subsystem. The log page is 4,096 bytes in size. There is one

Commands Supported and Effects data structure per Admin command and one Commands Supported and Effects data structure per I/O command based on:

a) the I/O Command Set selected in CC.CSS, if CC.CSS is not set to 110b; and
b) the Command Set Identifier field in CDW 14, if CC.CSS is set to 110b.

Figure 210: Commands Supported and Effects Log Page

Bytes	Description
03:00	**Admin Command Supported 0 (ACS0):** Contains the Commands Supported and Effects data structure (refer to Figure 211) for the Admin command with an opcode value of 0h.
07:04	**Admin Command Supported 1 (ACS1):** Contains the Commands Supported and Effects data structure (refer to Figure 211) for the Admin command with an opcode value of 1h.
…	…
1019:1016	**Admin Command Supported 254 (ACS254):** Contains the Commands Supported and Effects data structure (refer to Figure 211) for the Admin command with an opcode value of 254.
1023:1020	**Admin Command Supported 255 (ACS255):** Contains the Commands Supported and Effects data structure (refer to Figure 211) for the Admin command with an opcode value of 255.
1027:1024	**I/O Command Supported 0 (IOCS0):** Contains the Commands Supported and Effects data structure (refer to Figure 211) for the I/O command with an opcode value of 0h.
1031:1028	**I/O Command Supported 1 (IOCS1):** Contains the Commands Supported and Effects data structure (refer to Figure 211) for the I/O command with an opcode value of 1h.
…	…
2043:2040	**I/O Command Supported 254 (IOCS254):** Contains the Commands Supported and Effects data structure (refer to Figure 211) for the I/O command with an opcode value of 254.
2047:2044	**I/O Command Supported 255 (IOCS255):** Contains the Commands Supported and Effects data structure (refer to Figure 211) for the I/O command with an opcode value of 255.
4095:2048	Reserved

The Commands Supported and Effects data structure describes the overall possible effect of a command, including any optional features of the command.

Host software may take command effects into account when determining how to submit commands and actions to take after the command is complete. It is recommended that if a command may change a particular capability that host software re-enumerate and/or re-initialize the associated capability after the command is complete. For example, if a namespace capability change may occur, then host software is recommended to pause the use of the associated namespace, submit the command that may cause a namespace capability change and wait for its completion, and then re-issue the Identify command.

If the namespace is attached to multiple controllers, the host(s) associated with those controllers should coordinate their commands to meet the Command Submission and Execution requirements (refer to Figure 211). The details of this coordination are outside the scope of this specification.

Figure 211: Commands Supported and Effects Data Structure

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:20</td><td>Command Scope (CSP): This field defines the scope for the associated command. If the value of this field is 0h then no scope is reported.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>11:6</td><td>Reserved</td></tr>
<tr><td>5</td><td>NVM Subsystem Scope: If set to ‘1’, then the command performs actions that may impact the whole NVM subsystem. If cleared to ‘0’ and the CSP field is non-zero, then the command performs actions that does not impact the whole NVM subsystem.</td></tr>
<tr><td>4</td><td>Domain Scope: If set to ‘1’, then the command performs actions that may impact a single Domain. If cleared to ‘0’ and the CSP field is non-zero, then the command performs actions that does not impact a single Domain.</td></tr>
<tr><td>3</td><td>Endurance Group Scope: If set to ‘1’, then the command performs actions that may impact Endurance Groups. If cleared to ‘0’ and the CSP field is non-zero, then the command performs actions that does not impact a single Endurance Group.</td></tr>
<tr><td>2</td><td>NVM Set Scope: If set to ‘1’, then the command performs actions that may impact NVM Sets. If cleared to ‘0’ and the CSP field is non-zero, then the command performs actions that do not impact NVM Sets.</td></tr>
<tr><td>1</td><td>Controller Scope: If set to ‘1’, then the command performs actions that may impact controllers. If cleared to ‘0’ and the CSP field is non-zero, then the command performs actions that do not impact controllers.</td></tr>
<tr><td>0</td><td>Namespace Scope: If set to ‘1’, then the command performs actions that may impact namespaces. If cleared to ‘0’ and the CSP field is non-zero, then the command performs actions that do not impact namespaces.</td></tr>
</table></td></tr>
<tr><td>19</td><td>UUID Selection Supported: If set to ‘1’, then the controller supports selection of a UUID by this command (refer to section 8.25). If cleared to ‘0’, then the controller does not support selection of a UUID by this command.</td></tr>
<tr><td>18:16</td><td>Command Submission and Execution (CSE): This field defines the command submission and execution recommendations for the associated command.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>No command submission or execution restriction</td></tr>
<tr><td>001b</td><td>The command associated with this structure should only be submitted when there is no other outstanding command affecting the same namespace and another command should not be submitted that affects the same namespace until this command is complete.</td></tr>
<tr><td>010b</td><td>The command associated with this structure should only be submitted when there is no other outstanding command that affects any namespace and another command should not be submitted that affects any namespace until this command is complete.</td></tr>
<tr><td>011b to 111b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>15:05</td><td>Reserved</td></tr>
<tr><td>04</td><td>Controller Capability Change (CCC): If this bit is set to ‘1’, then this command may change controller capabilities. If this bit is cleared to ‘0’, then this command does not modify controller capabilities. Controller capability changes include a firmware update that changes the capabilities reported in the CAP property.</td></tr>
<tr><td>03</td><td>Namespace Inventory Change (NIC): If this bit is set to ‘1’, then this command may change the number of namespaces or capabilities for multiple namespaces. If this bit is cleared to ‘0’, then this command does not modify the number of namespaces or capabilities for multiple namespaces. Namespace inventory changes include adding or removing namespaces.</td></tr>
<tr><td>02</td><td>Namespace Capability Change (NCC): If this bit is set to ‘1’, then this command may change the capabilities of a single namespace. If this bit is cleared to ‘0’, then this command does not modify any namespace capabilities for the specified namespace. Namespace capability changes include a logical format change.</td></tr>
</table>

Figure 211: Commands Supported and Effects Data Structure

Bits	Description
01	**Logical Block Content Change (LBCC):** If this bit is set to '1', then this command may modify user data content in one or more namespaces. If this bit is cleared to '0', then this command does not modify user data content in any namespace. User data content changes include a write to user data. NOTE: This field applies to all user data content changes. The original name has been retained for historical continuity.
00	**Command Supported (CSUPP):** If this bit is set to '1', then this command is supported by the controller. If this bit is cleared to '0', then this command is not supported by the controller and all other fields in this structure shall be cleared to 0h.

5.16.1.7 Device Self-test (Log Identifier 06h)

This log page is used to indicate:

a) the status of any device self-test operation in progress and the percentage complete of that operation; and
b) the results of the last 20 device self-test operations.

The Self-test Result Data Structure contained in the Newest Self-test Result Data Structure field is always the result of the last completed or aborted self-test operation. The next Self-test Result Data Structure field in the Device Self-test log page contains the results of the second newest self-test operation and so on. If fewer than 20 self-test operations have completed or been aborted, then the Device Self-test Status field shall be set to Fh in the unused Self-test Result Data Structure fields and all other fields in that Self-test Result Data Structure are ignored.

Figure 212: Device Self-test Log Page

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>00</td><td>Current Device Self-Test Operation: This field defines the current device self-test operation.

Bits 7:4 are reserved.

Bits 3:0 indicates the status of the current device self-test operation as defined in the following table. If a device self-test operation is in process (i.e., this field is set to 1h or 2h), then the controller shall not set this field to 0h until a new Self-test Result Data Structure is created (i.e., if a device self-test operation completes or is aborted, then the controller shall create a Self-test Result Data Structure prior to setting this field to 0h).
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0h</td><td>No device self-test operation in progress</td></tr>
<tr><td>1h</td><td>Short device self-test operation in progress</td></tr>
<tr><td>2h</td><td>Extended device self-test operation in progress</td></tr>
<tr><td>3h to Dh</td><td>Reserved</td></tr>
<tr><td>Eh</td><td>Vendor specific</td></tr>
<tr><td>Fh</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>01</td><td>Current Device Self-Test Completion: This field defines the completion status of the current device self-test.

Bit 7 is reserved.

Bits 6:0 indicates the percentage of the device self-test operation that is complete (e.g., a value of 25 indicates that 25% of the device self-test operation is complete and 75% remains to be tested). If bits 3:0 in the Current Device Self-Test Operation field are cleared to 0h (indicating there is no device self-test operation in progress), then this field is ignored.</td></tr>
<tr><td>03:02</td><td>Reserved</td></tr>
<tr><td>31:04</td><td>Newest Self-test Result Data Structure (refer to Figure 213)</td></tr>
<tr><td>59:32</td><td>2nd newest Self-test Result Data Structure (refer to Figure 213)</td></tr>
<tr><td>…</td><td>…</td></tr>
<tr><td>535:508</td><td>19th newest Self-test Result Data Structure (refer to Figure 213)</td></tr>
</table>

Figure 212: Device Self-test Log Page

Bytes	Description
563:536	20th newest Self-test Result Data Structure (refer to Figure 213)

Figure 213: Self-test Result Data Structure

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>00</td><td>
Device Self-test Status: This field indicates the device self-test code and the status of the operation.

Bits 7:4 indicates the Self-test Code value that was specified in the Device Self-test command that started the device self-test operation that this Self-test Result Data Structure describes.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0h</td><td>Reserved</td></tr>
<tr><td>1h</td><td>Short device self-test operation</td></tr>
<tr><td>2h</td><td>Extended device self-test operation</td></tr>
<tr><td>3h to Dh</td><td>Reserved</td></tr>
<tr><td>Eh</td><td>Vendor specific</td></tr>
<tr><td>Fh</td><td>Reserved</td></tr>
</table>
Bits 3:0 indicates the result of the device self-test operation that this Self-test Result Data Structure describes.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0h</td><td>Operation completed without error</td></tr>
<tr><td>1h</td><td>Operation was aborted by a Device Self-test command</td></tr>
<tr><td>2h</td><td>Operation was aborted by a Controller Level Reset</td></tr>
<tr><td>3h</td><td>Operation was aborted due to a removal of a namespace from the namespace inventory</td></tr>
<tr><td>4h</td><td>Operation was aborted due to the processing of a Format NVM command</td></tr>
<tr><td>5h</td><td>A fatal error or unknown test error occurred while the controller was executing the device self-test operation and the operation did not complete</td></tr>
<tr><td>6h</td><td>Operation completed with a segment that failed and the segment that failed is not known</td></tr>
<tr><td>7h</td><td>Operation completed with one or more failed segments and the first segment that failed is indicated in the Segment Number field</td></tr>
<tr><td>8h</td><td>Operation was aborted for unknown reason</td></tr>
<tr><td>9h</td><td>Operation was aborted due to a sanitize operation</td></tr>
<tr><td>Ah to Eh</td><td>Reserved</td></tr>
<tr><td>Fh</td><td>Entry not used (does not contain a test result)</td></tr>
</table>
</td></tr>
<tr><td>01</td><td>Segment Number: This field indicates the segment number (refer to section 8.6) where the first self-test failure occurred. If Device Self-test Status field bits [3:0] are not set to 7h, then this field should be ignored.</td></tr>
<tr><td>02</td><td>
Valid Diagnostic Information: This field indicates the diagnostic failure information that is reported.

Bits 7:4 are reserved.

Bit 3 (SC Valid): If set to ‘1’, then the contents of Status Code field are valid. If cleared to ‘0’, then the contents of the Status Code field are invalid.

Bit 2 (SCT Valid): If set to ‘1’, then the contents of the Status Code Type field are valid. If cleared to ‘0’, then the contents of the Status Code Type field are invalid.

Bit 1 (FLBA Valid): If set to ‘1’, then the contents of the Failing LBA field are valid. If cleared to ‘0’, then the contents of the Failing LBA field are invalid.

Bit 0 (NSID Valid): If set to ‘1’, then the contents of the Namespace Identifier field are valid. If cleared to ‘0’, then the contents of the Namespace Identifier field are invalid.
</td></tr>
<tr><td>03</td><td>Reserved</td></tr>
</table>

Figure 213: Self-test Result Data Structure

Bytes	Description
11:04	**Power On Hours (POH):** This field indicates the number of power-on hours at the time the device self-test operation was completed or aborted. This does not include time that the controller was powered and in a low power state condition.
15:12	**Namespace Identifier (NSID):** This field indicates the namespace that the Failing LBA occurred on. The contents of this field are valid only when the NSID Valid bit is set to '1'.
23:16	**Failing LBA:** This field is I/O Command Set specific and is described in the applicable I/O Command Set specification. NOTE: The original field name has been retained for historical continuity.
24	**Status Code Type:** This field may contain additional information related to errors or conditions. Bits 7:3 are reserved. Bits 2:0 may contain additional information relating to errors or conditions that occurred during the device self-test operation represented in the same format used in the Status Code Type field of the completion queue entry (refer to Figure 93). The contents of this field are valid only when the SCT Valid bit is set to '1'.
25	**Status Code:** This field may contain additional information relating to errors or conditions that occurred during the device self-test operation represented in the same format used in the Status Code field of the completion queue entry (refer to section 3.3.3.2.1). The contents of this field are valid only when the SC Valid bit is set to '1'.
27:26	Vendor Specific

5.16.1.8 Telemetry Host-Initiated (Log Identifier 07h)

This log consists of a header describing the log and zero or more Telemetry Data Blocks (refer to section 8.24). All Telemetry Data Blocks are 512 bytes in size. The controller shall initiate a capture of the controller's internal controller state to this log if the controller processes a Get Log Page command for this log with the Create Telemetry Host-Initiated Data bit set to '1' in the Log Specific Parameter field. If the host specifies a Log Page Offset Lower value that is not a multiple of 512 bytes in the Get Log Page command for this log, then the controller shall abort the command with a status code of Invalid Field in Command. This log page is global to the controller or global to the NVM subsystem.

The Log Specific Parameter field in Command Dword 10 (refer to Figure 197) for this log page is defined in Figure 214.

Figure 214: Telemetry Host-Initiated Log Specific Parameter Field

Bits	Description
14:09	Reserved
08	**Create Telemetry Host-Initiated Data:** If set to '1', then the controller shall capture the Telemetry Host-Initiated Data representing the internal state of the controller at the time the associated Get Log Page command is processed. If cleared to '0', then the controller shall not update the Telemetry Host-Initiated Data. The Host-Initiated Data shall not change until the controller processes: a) a subsequent Telemetry Host-Initiated Log with this bit set to '1'; b) a Firmware Commit command; or c) a power on reset.

The Telemetry Host-Initiated Data consists of:

a) Three areas, if bit 6 of the Log Page Attributes field is cleared to '0': Telemetry Host-Initiated Data Area 1, Telemetry Host-Initiated Data Area 2, and Telemetry Host-Initiated Data Area 3; or
b) Four areas, if bit 6 of the Log Page Attributes field is set to '1': Telemetry Host-Initiated Data Area 1, Telemetry Host-Initiated Data Area 2, Telemetry Host-Initiated Data Area 3 and Telemetry Host-Initiated Data Area 4.

All areas start at Telemetry Host-Initiated Data Area Block 1. The last block of each area is indicated in Telemetry Host-Initiated Data Area *y* Last Block, respectively. The telemetry data captured and its size is implementation dependent.

The size of the log page is variable and:

- If bit 6 is cleared to '0' in the Log Page Attributes field, the size may be calculated using the Telemetry Host-Initiated Data Area 3 Last Block field.
- If bit 6 of the Log Page Attributes field is set to '1' and the Extended Telemetry Data Area 4 Supported (ETDAS) field is set to 1h in the Host Behavior Support feature (refer to section 5.21.1.22), then the size of the log page may be calculated using the Telemetry Host-Initiated Data Area 4 Last Block field.
- If bit 6 of the Log Page Attributes field is set to '1' and the Extended Telemetry Data Area 4 Supported (ETDAS) field is cleared to 0h in the Host Behavior Support feature (refer to section 5.21.1.22), then the size of the log page may be calculated using the Telemetry Host-Initiated Data Area 3 Last Block field.

The controller shall return data for all blocks requested:

- If bit 6 of the Log Page Attributes field is cleared to '0', then the data beyond the last block in Telemetry Host-Initiated Data Area 3 Last Block is undefined.
- If bit 6 of the Log Page Attributes field is set to '1' and the Extended Telemetry Data Area 4 Supported (ETDAS) field is set to 1h in the Host Behavior Support feature, then the data beyond the last block in Telemetry Host-Initiated Data Area 4 Last Block is undefined.
- If bit 6 of the Log Page Attributes field is set to '1' and the Extended Telemetry Data Area 4 Supported (ETDAS) field is cleared to 0h in the Host Behavior Support feature, then the data beyond the last block in Telemetry Host-Initiated Data Area 3 Last Block is undefined.

If the host requests a data transfer that is not a multiple of 512 bytes, then the controller shall return an error of Invalid Field in Command.

Figure 215: Telemetry Host-Initiated Log Page

Bytes	Description
00	**Log Identifier:** This field shall be set to 07h.
04:01	Reserved
07:05	**IEEE OUI Identifier (IEEE):** Contains the Organization Unique Identifier (OUI) for the controller vendor that is able to interpret the data. If cleared to 0h, no IEEE OUI Identifier is present. The OUI shall be a valid IEEE/RAC assigned identifier that is registered at http://standards.ieee.org/develop/regauth/oui/public.html.
09:08	**Telemetry Host-Initiated Data Area 1 Last Block:** Contains the value of the last block of Telemetry Host-Initiated Data Area 1. If the Telemetry Host-Initiated Data Area 1 does not contain data, then this field shall be cleared to 0h. If this field is not 0h, then Telemetry Host-Initiated Data Area 1 begins at block 1h and ends at the block indicated in this field.
11:10	**Telemetry Host-Initiated Data Area 2 Last Block:** Contains the value of the last block of Telemetry Host-Initiated Data Area 2. This value shall be greater than or equal to the value in the Telemetry Host-Initiated Data Area 1 Last Block field. If this field is not 0h, then Telemetry Host-Initiated Data Area 2 begins at block 1h and ends at the block indicated in this field.
13:12	**Telemetry Host-Initiated Data Area 3 Last Block:** Contains the value of the last block of Telemetry Host-Initiated Data Area 3. This value shall be greater than or equal to the value in the Telemetry Host-Initiated Data Area 2 Last Block field. If this field is not 0h, then Telemetry Host-Initiated Data Area 3 begins at block 1h and ends at the block contained in this field.
15:14	Reserved

Figure 215: Telemetry Host-Initiated Log Page

Bytes	Description
19:16	**Telemetry Host-Initiated Data Area 4 Last Block:** Contains the value of the last block of Telemetry Host-Initiated Data Area 4. If bit 6 of the Log Page Attributes field is set to '1', then this value shall be greater than or equal to the value in the Telemetry Host-Initiated Data Area 3 Last Block field. If this field is not 0h, then Telemetry Host-Initiated Data Area 4 begins at block 1h and ends at the block contained in this field.
380:20	Reserved
381	**Telemetry Host-Initiated Data Generation Number:** Contains a value that is incremented each time the controller captures its internal controller state for this log page. If the value of this field is FFh, then the field shall be cleared to 0h when incremented (i.e., rolls over to 0h).
382	**Telemetry Controller-Initiated Data Available:** Contains the value of Telemetry Controller-Initiated Data Available field in the Telemetry Controller-Initiated log (refer to Figure 216).
383	**Telemetry Controller-Initiated Data Generation Number:** Contains the value of the Telemetry Controller-Initiated Data Generation Number field in the Telemetry Controller-Initiated log (refer to Figure 216).
511:384	**Reason Identifier:** Contains a vendor specific identifier that describes the operating conditions of the controller at the time of capture. The Reason Identifier field should provide an identification of unique operating conditions of the controller.
1023:512	**Telemetry Host-Initiated Data Block 1:** Contains Telemetry Data Block 1 for the Telemetry Host-Initiated Log.
1535:1024	**Telemetry Host-Initiated Data Block 2:** Contains Telemetry Data Block 2 for the Telemetry Host-Initiated Log.
…	…
(*n**512)+511:(*n**512)	**Telemetry Host-Initiated Data Block *n*:** Contains Telemetry Data Block *n* for the Telemetry Host-Initiated Log.

5.16.1.9 Telemetry Controller-Initiated (Log Identifier 08h)

This log consists of a header describing the log and zero or more Telemetry Data Blocks (refer to section 8.23). All Telemetry Data Blocks are 512 bytes in size. This log is a controller initiated capture of the controller's internal state. The Telemetry Controller-Initiated Data for Data Area 1 through Data Area 3 shall persist across all resets. The Telemetry Controller-Initiated Data for Data Area 4 may persist across controller resets. If the host specifies a Log Page Offset Lower value that is not a multiple of 512 bytes in the Get Log Page command for this log, then the controller shall return an error of Invalid Field in Command. This log page is global to the controller.

The Telemetry Controller-Initiated Data consists of:

a) three areas, if bit 6 of the Log Page Attributes field is cleared to '0': Telemetry Controller-Initiated Data Area 1, Telemetry Controller-Initiated Data Area 2, and Telemetry Controller-Initiated Data Area 3; or
b) four areas, if bit 6 of the Log Page Attributes field is set to '1': Telemetry Controller-Initiated Data Area 1, Telemetry Controller-Initiated Data Area 2, Telemetry Controller-Initiated Data Area 3 and Telemetry Controller-Initiated Data Area 4.

All areas start at Telemetry Controller-Initiated Data Area Block 1. The last block of each area is indicated in the Telemetry Controller-Initiated Data Area y Last Block, respectively. The telemetry data captured and its size is implementation dependent.

The size of the log page is variable and:

- If bit 6 is cleared to '0' in the Log Page Attributes field, the size may be calculated using the Telemetry Controller-Initiated Data Area 3 Last Block field.
- If bit 6 of the Log Page Attributes field is set to '1' and the Extended Telemetry Data Area 4 Supported (ETDAS) field is set to 1h in the Host Behavior Support feature (refer to section

5.27.1.18), then the size of the log page may be calculated using the Telemetry Controller-Initiated Data Area 4 Last Block field.

- If bit 6 of the Log Page Attributes field is set to '1' and the Extended Telemetry Data Area 4 Supported (ETDAS) field is cleared to 0h in the Host Behavior Support feature (refer to section 5.27.1.18), then the size of the log page may be calculated using the Telemetry Controller-Initiated Data Area 3 Last Block field.

The controller shall return data for all blocks requested:

- If bit 6 of the Log Page Attributes field is cleared to '0', then the data beyond the last block in Telemetry Controller-Initiated Data Area 3 Last Block is undefined.
- If bit 6 of the Log Page Attributes field is set to '1', then the data beyond the last block in Telemetry Controller-Initiated Data Area 4 Last Block is undefined.
- If bit 6 of the Log Page Attributes field is set to '1' and the Extended Telemetry Data Area 4 Supported (ETDAS) field is cleared to 0h in the Host Behavior Support feature, then the data beyond the last block in Telemetry Controller-Initiated Data Area 3 Last Block is undefined.

If the host requests a data transfer that is not a multiple of 512 bytes, then the controller shall abort the command with the status code of Invalid Field in Command.

Figure 216: Telemetry Controller-Initiated Log Page

Bytes	Description
00	**Log Identifier:** This field shall be set to 08h.
04:01	Reserved
07:05	**IEEE OUI Identifier (IEEE):** Contains the Organization Unique Identifier (OUI) for the controller vendor that is able to interpret the data. If cleared to 0h, no IEEE OUI Identifier is present. The OUI shall be a valid IEEE/RAC assigned identifier that is registered at http://standards.ieee.org/develop/regauth/oui/public.html.
09:08	**Telemetry Controller-Initiated Data Area 1 Last Block:** Contains the value of the last block of Telemetry Controller-Initiated Data Area 1. If the Telemetry Controller-Initiated Data Area 1 does not contain data, then this field shall be cleared to 0h. If this field is not 0h, then Telemetry Controller-Initiated Data Area 1 begins at block 1 and ends at the block indicated in this field.
11:10	**Telemetry Controller-Initiated Data Area 2 Last Block:** Contains the value of the last block of Telemetry Controller-Initiated Data Area 2. This value shall be greater than or equal to the value in the Telemetry Controller-Initiated Data Area 1 Last Block field. If this field is not 0h, then Telemetry Controller-Initiated Data Area 2 begins at block 1h and ends at the block indicated in this field.
13:12	**Telemetry Controller-Initiated Data Area 3 Last Block:** Contains the value of the last block of Telemetry Controller-Initiated Data Area 3. This value shall be greater than or equal to the value in the Telemetry Controller-Initiated Data Area 2 Last Block field. If this field is not 0h, then Telemetry Controller-Initiated Data Area 3 begins at block 1h and ends at the block indicated in this field.
15:14	Reserved
19:16	**Telemetry Controller-Initiated Data Area 4 Last Block:** Contains the value of the last block of Telemetry Controller-Initiated Data Area 4. If bit 6 of the Log Page Attributes field is set to '1', then this value shall be greater than or equal to the value in the Telemetry Controller-Initiated Data Area 3 Last Block field. If this field is not 0h, then Telemetry Controller-Initiated Data Area 4 begins at block 1h and ends at the block contained in this field.
381:20	Reserved
382	**Telemetry Controller-Initiated Data Available:** If this field is cleared to 0h, the log does not contain saved internal controller state. If this field is set to 1h, the log contains saved internal controller state. If this field is set to 1h, it shall not be cleared to 0h until a Get Log Page command with Retain Asynchronous Event bit cleared to '0' for the Telemetry Controller-Initiated log completes successfully. This value is persistent across power states and reset.

Figure 216: Telemetry Controller-Initiated Log Page

Bytes	Description
	Other values are reserved.
383	**Telemetry Controller-Initiated Data Generation Number:** Contains a value that is incremented each time the controller initiates a capture of its internal controller state into the Telemetry Controller-Initiated Data Blocks. If the value of this field is FFh, then the field shall be cleared to 0h when incremented (i.e., rolls over to 0h). This field is persistent across power cycles.
511:384	**Reason Identifier:** Contains a vendor specific identifier that describes the operating conditions of the controller at the time of capture. The Controller-Initiated Reason Identifier field should provide an identification of unique operating conditions of the controller.
1023:512	**Telemetry Controller-Initiated Data Block 1:** Contains Telemetry Data Block 1 for the Telemetry Controller -Initiated Log captured at a vendor specific time.
1535:1024	**Telemetry Controller-Initiated Data Block 2:** Contains Telemetry Data Block 2 for the Telemetry Controller -Initiated Log captured at a vendor specific time.
…	…
(*n**512)+511:(*n**512)	**Telemetry Controller-Initiated Data Block *n*:** Contains Telemetry Data Block *n* for the Telemetry Controller-Initiated log captured at a vendor specific time.

5.16.1.10 Endurance Group Information (Log Identifier 09h)

This log page is used to provide endurance information based on the Endurance Group (refer to section 3.2.3). An Endurance Group contains capacity that may be allocated to zero or more NVM Sets. Capacity that has not been allocated to an NVM Set is unallocated Endurance Group capacity. The information provided is over the life of the Endurance Group. The Endurance Group Identifier is specified in the Log Specific Identifier field in Command Dword 11 of the Get Log Page command. The log page is 512 bytes in size.

Figure 217: Endurance Group Information Log Page

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>00</td><td>Critical Warning: This field indicates critical warnings for the state of the Endurance Group. Each bit corresponds to a critical warning type; multiple bits may be set to ‘1’. If a bit is cleared to ‘0’, then that critical warning does not apply. Critical warnings may result in an asynchronous event notification to the host. Bits in this field represent the state at the time the Get Log Page command is processed and may not reflect the state at the time a related asynchronous event notification, if any, occurs or occurred.

If a bit is set to ‘1’ in all Endurance Groups in the NVM subsystem, then the corresponding bit shall be set to ‘1’ in the Critical Warning field of the SMART / Health Information log page (refer to Figure 207).
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>7:4</td><td>Reserved</td></tr>
<tr><td>3</td><td>If set to ‘1’, then all namespaces in the Endurance Group have been placed in read only mode for reasons other than a change in the write protect state of the namespace. The controller shall not set this bit to '1' if the read-only condition on the Endurance Group is a result of a change in the write protection state of all namespaces in the Endurance Group.</td></tr>
<tr><td>2</td><td>If set to ‘1’, then the Endurance Group reliability has been degraded due to significant media related errors or any internal error that degrades NVM subsystem reliability.</td></tr>
<tr><td>1</td><td>Reserved</td></tr>
<tr><td>0</td><td>If set to ‘1’, then the available spare capacity of the Endurance Group has fallen below the threshold.</td></tr>
</table>
</td></tr>
</table>

Figure 217: Endurance Group Information Log Page

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>01</td><td>Endurance Group Features (EGFEAT): This field defines features of the Endurance Group.
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>7:1</td><td>Reserved</td></tr>
<tr><td>0</td><td>Endurance Group Media (EGRMEDIA): If set to ‘1’ indicates that the Endurance Group stores data on rotational media (refer to section 8.20). If cleared to ‘0’, then the Endurance Group does not store data on rotational media.</td></tr>
</table></td></tr>
<tr><td>02</td><td>Reserved</td></tr>
<tr><td>03</td><td>Available Spare: Contains a normalized percentage (0% to 100%) of the remaining spare capacity available for the Endurance Group.</td></tr>
<tr><td>04</td><td>Available Spare Threshold: If the Available Spare falls below the threshold indicated in this field, an asynchronous event completion may occur. The value is indicated as a normalized percentage (0% to 100%). The values 101 to 255 are reserved.</td></tr>
<tr><td>05</td><td>Percentage Used: Contains a vendor specific estimate of the percentage of life used for the Endurance Group based on the actual usage and the manufacturer’s prediction of NVM life. A value of 100 indicates that the estimated endurance of the NVM in the Endurance Group has been consumed, but may not indicate an NVM failure. The value is allowed to exceed 100. Percentages greater than 254 shall be represented as 255. This value shall be updated once per power-on hour when the controller is not in a sleep state.

Refer to the JEDEC JESD218A standard for SSD device life and endurance measurement techniques.</td></tr>
<tr><td>07:06</td><td>Domain Identifier: This field indicates the identifier of the domain that contains this Endurance Group. If the NVM subsystem supports multiple domains, this field shall be set to a non-zero value. If cleared to 0h, the NVM subsystem does not support multiple domains.</td></tr>
<tr><td>31:08</td><td>Reserved</td></tr>
<tr><td>47:32</td><td>Endurance Estimate: This field is an estimate of the total number of data bytes that may be written to the Endurance Group over the lifetime of the Endurance Group assuming a write amplification of 1 (i.e., no increase in the number of write operations performed by the device beyond the number of write operations requested by a host). This value is reported in billions (i.e., a value of 1 corresponds to 1,000,000,000 bytes written) and is rounded up (e.g., one indicates the number of bytes written is from 1 to 1,000,000,000, three indicates the number of bytes written is from 2,000,000,001 to 3,000,000,000).

A value of 0h indicates that the controller does not report an Endurance Estimate.</td></tr>
<tr><td>63:48</td><td>Data Units Read: Contains the total number of data bytes that have been read from the Endurance Group. This value does not include controller reads due to internal operations such as garbage collection. This value is reported in billions (i.e., a value of 1 corresponds to 1,000,000,000 bytes read) and is rounded up (e.g., one indicates the number of bytes read is from 1 to 1,000,000,000, three indicates the number of bytes read is from 2,000,000,001 to 3,000,000,000).

A value of 0h indicates that the controller does not report the number of Data Units Read.</td></tr>
<tr><td>79:64</td><td>Data Units Written: Contains the total number of data bytes that have been written to the Endurance Group. This value does not include controller writes due to internal operations such as garbage collection. This value is reported in billions (i.e., a value of 1 corresponds to 1,000,000,000 bytes written) and is rounded up (e.g., one indicates the number of bytes written is from 1 to 1,000,000,000, three indicates the number of bytes written is from 2,000,000,001 to 3,000,000,000).

A value of 0h indicates that the controller does not report the number of Data Units Written.</td></tr>
<tr><td>95:80</td><td>Media Units Written: Contains the total number of data bytes that have been written to the Endurance Group including both host and controller writes (e.g., garbage collection). This value is reported in billions (i.e., a value of 1 corresponds to 1,000,000,000 bytes written) and is rounded up (e.g., one indicates the number of bytes written is from 1 to 1,000,000,000, three indicates the number of bytes written is from 2,000,000,001 to 3,000,000,000).

A value of 0h indicates that controller does not report the number of Media Units Written.</td></tr>
</table>

Figure 217: Endurance Group Information Log Page

Bytes	Description
111:96	**Host Read Commands:** Contains the number of Endurance Group Host Read Commands completed by the controller. Refer to the specific I/O Command Set specification for the list of Endurance Group Host Read Commands that affect this field.
127:112	**Host Write Commands:** Contains the number of User Data Out Commands completed by the controller. Refer to the specific I/O Command Set specification for the list of User Data Out Commands that affect this field.
143:128	**Media and Data Integrity Errors:** Contains the number of occurrences where the controller detected an unrecovered data integrity error for the Endurance Group. Errors such as uncorrectable ECC, CRC checksum failure, or LBA tag mismatch are included in this field.
159:144	**Number of Error Information Log Entries:** Contains the number of Error Information Log Entries over the life of the controller for the Endurance Group.
175:160	**Total Endurance Group Capacity (TEGCAP):** This field indicates the total NVM capacity in this Endurance Group. The value is in bytes. If this field is cleared to 0h, the NVM subsystem does not report the total NVM capacity in this Endurance Group.
191:176	**Unallocated Endurance Group Capacity (UEGCAP):** This field indicates the unallocated NVM capacity in this Endurance Group. The value is in bytes. If this field is cleared to 0h, the NVM subsystem does not report the unallocated NVM capacity in this Endurance Group.
511:192	Reserved

5.16.1.11 Predictable Latency Per NVM Set (Log Identifier 0Ah)

This log page may be used to determine the current window for the specified NVM Set when Predictable Latency Mode is enabled and any events that have occurred for the specified NVM Set. There is one log page for each NVM Set when Predictable Latency Mode is supported. Command Dword 11 (refer to Figure 198) specifies the NVM Set for which the log page is to be returned. The log page is 512 bytes in size.

The log page indicates typical values and reliable estimates for attributes associated with the Deterministic Window and the Non-Deterministic Window of the specified NVM Set. The Typical, Maximum, and Minimum values are static and worst-case values over the lifetime of the NVM subsystem.

After the controller successfully completes a read of this log page with Retain Asynchronous Event bit cleared to ‘0’, then reported events are cleared to ‘0’ for the specified NVM Set and the field corresponding to the specified NVM Set is cleared to ‘0’ in the Predictable Latency Event Aggregate log page. Co-ordination between two or more hosts is beyond the scope of this specification.

Figure 218: Predictable Latency Per NVM Set Log Page

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>00</td><td>Status: This field indicates the status of the specified NVM Set.
Bits 7:3 are reserved.
Bits 2:0 indicate the window for the NVM Set when Predictable Latency Mode is enabled.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>Not used (Predictable Latency Mode not enabled)</td></tr>
<tr><td>001b</td><td>Deterministic Window (DTWIN)</td></tr>
<tr><td>010b</td><td>Non-Deterministic Window (NDWIN)</td></tr>
<tr><td>011b to 111b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>01</td><td>Reserved</td></tr>
</table>

Figure 218: Predictable Latency Per NVM Set Log Page

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>03:02</td><td>Event Type: This field specifies the event(s) that occurred for the NVM Set indicated. Multiple bits may be set to ‘1’. All bits are cleared to ‘0’ after the log page is read with Retain Asynchronous Event bit cleared to ‘0’.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>00</td><td>DTWIN Reads Warning</td></tr>
<tr><td>01</td><td>DTWIN Writes Warning</td></tr>
<tr><td>02</td><td>DTWIN Time Warning</td></tr>
<tr><td>03 to 13</td><td>Reserved</td></tr>
<tr><td>14</td><td>Autonomous transition from DTWIN to NDWIN due to typical or maximum value exceeded</td></tr>
<tr><td>15</td><td>Autonomous transition from DTWIN to NDWIN due to Deterministic Excursion</td></tr>
</table></td></tr>
<tr><td>31:04</td><td>Reserved</td></tr>
<tr><th colspan="2">Typical, Maximum, and Minimum Values</th></tr>
<tr><td>39:32</td><td>DTWIN Reads Typical: Indicates the typical number of 4 KiB random reads that may be performed in the Deterministic Window. Refer to section 8.16.</td></tr>
<tr><td>47:40</td><td>DTWIN Writes Typical: Indicates the typical number of writes in units of the Optimal Write Size that may be performed in the Deterministic Window. Refer to section 8.16.</td></tr>
<tr><td>55:48</td><td>DTWIN Time Maximum: Indicates the maximum time in milliseconds that the NVM Set is able to remain in a Deterministic Window before entering a Non-Deterministic Window. Refer to section 8.16.</td></tr>
<tr><td>63:56</td><td>NDWIN Time Minimum High: Indicates the minimum time in milliseconds that the NVM Set needs to remain in the Non-Deterministic Window before entering a Deterministic Window. This is the time necessary to prepare for remaining in the Deterministic Window for DTWIN Time Maximum. Refer to section 8.16.</td></tr>
<tr><td>71:64</td><td>NDWIN Time Minimum Low: Indicates the minimum time in milliseconds that the NVM Set needs to remain in the Non-Deterministic Window before entering a Deterministic Window. This is regardless of the amount of time spent in the previous Deterministic Window. Refer to section 8.16.</td></tr>
<tr><td>127:72</td><td>Reserved</td></tr>
<tr><th colspan="2">Reliable Estimates</th></tr>
<tr><td>135:128</td><td>DTWIN Reads Estimate: Indicates a reliable estimate of the number of 4 KiB random reads remaining in the current Deterministic Window, if applicable. This value decrements from DTWIN Reads Typical to 0h based on host read activity and operating conditions. Refer to section 8.16.1.</td></tr>
<tr><td>143:136</td><td>DTWIN Writes Estimate: Indicates a reliable estimate of the number of writes in units of the Optimal Write Size remaining in the current Deterministic Window, if applicable. This value decrements from DTWIN Writes Typical to 0h based on host write activity and operating conditions. Refer to section 8.16.1.</td></tr>
<tr><td>151:144</td><td>DTWIN Time Estimate: Indicates a reliable estimate of the time in milliseconds remaining in the current Deterministic Window, if applicable. Refer to section 8.16.1.</td></tr>
<tr><td>511:152</td><td>Reserved</td></tr>
</table>

5.16.1.12 Predictable Latency Event Aggregate (Log Identifier 0Bh)

This log page indicates if a Predictable Latency Event (refer to section 8.16) has occurred for a particular NVM Set. If a Predictable Latency Event has occurred, the details of the particular event are included in the Predictable Latency Per NVM Set log page for that NVM Set. An asynchronous event is generated when an entry for an NVM Set is newly added to this log page.

This log page shall not contain an entry (i.e., an NVM Set Identifier) that is cleared to 0h.

If there is an enabled Predictable Latency Event pending for an NVM Set, then the Predictable Latency Event Aggregate log page includes an entry for that NVM Set. The log page is an ordered list by NVM Set Identifier. For example, if Predictable Latency Events are pending for NVM Set 27, 13, and 17, then the log page shall have entries in numerical order of 13, 17, and 27. A particular NVM Set is removed from this log page after the Get Log Page command is completed successfully with the Retain Asynchronous Event bit cleared to ‘0’ for the Predictable Latency Per NVM Set log page for that NVM Set.

The log page size is limited by the NVM Set Identifier Maximum value reported in the Identify Controller data structure (refer to Figure 275). If the host reads beyond the end of the log page, zeroes are returned. The log page is defined in Figure 219.

Figure 219: Predictable Latency Event Aggregate Log Page

Bytes	Description
07:00	**Number of Entries:** This field indicates the number of entries in the list. The maximum number of entries in the list corresponds to the NVM Set Identifier Maximum field reported in the Identify Controller data structure. A value of 0h indicates there are no entries in the list.
09:08	**Entry 1:** Indicates the NVM Set that has a Predictable Latency Event pending that has the numerically smallest NVM Set Identifier.
11:10	**Entry 2:** Indicates the NVM Set that has a Predictable Latency Event pending that has the second numerically smallest NVM Set Identifier, if any.
13:12	**Entry 3:** Indicates the NVM Set that has a Predictable Latency Event pending that has the third numerically smallest NVM Set Identifier, if any.
15:14	**Entry 4:** Indicates the NVM Set that has a Predictable Latency Event pending that has the fourth numerically smallest NVM Set Identifier, if any.
…	…

5.16.1.13 Asymmetric Namespace Access (Log Identifier 0Ch)

This log consists of a header describing the log and descriptors containing the asymmetric namespace access information for ANA Groups (refer to section 8.1.2) that contain namespaces that are attached to the controller processing the command. If ANA Reporting (refer to section 8.1) is supported, this log page is supported. ANA Group Descriptors shall be returned in ascending ANA Group Identifier order.

If the Index Offset Supported bit is cleared to ‘0’ in the LID Support and Effects data structure for this log page (refer to Figure 204), then:

- if the RGO bit is cleared to ‘0’ in Command Dword 10, then the LPOL field in Command Dword 12 and the LPOU field in Command Dword 13 of the Get Log Page command should be cleared to 0h.

If the Index Offset Supported bit is set to ‘1’ in the LID Supported and Effects data structure for this log page (refer to Figure 204), then:

- the entry data structure that is indexed is an ANA Group Descriptor (e.g., specifying an index offset of 2 returns this log page starting at the offset of ANA Group Descriptor 1).

If the host performs multiple Get Log Page commands to read the ANA log page (e.g., using the LPOL field or the LPOU field), the host should re-read the header of the log page and ensure that the Change Count field in the Asymmetric Namespace Access log matches the original value read. If it does not match, then the data captured is not consistent and the ANA log page should be re-read.

The Log Specific Parameter field in Command Dword 10 (refer to Figure 197) for this log page is defined in Figure 220.

Figure 220: Asymmetric Namespace Access Log Specific Parameter Field

Bits	Description
14:09	Reserved
08	**Return Groups Only (RGO):** If set to ‘1’, then the controller shall return ANA Group Descriptors with the Number of NSID Values field in each ANA Group Descriptor cleared to 0h (i.e., no Namespace Identifiers are returned). If cleared to ‘0’, then the controller shall return ANA Group Descriptors that contain the Namespace Identifiers of attached namespaces that are members of the ANA Group described by that ANA Group Descriptor and the Number of NSID Values field set to the number of Namespace Identifier values in that ANA Group Descriptor.

Figure 221: Asymmetric Namespace Access Log Page

Bytes	Description
07:00	**Change Count:** This field contains a 64-bit incrementing Asymmetric Namespace Access log change count, indicating an identifier for this set of asymmetric namespace access information. The count starts at 0h following a Controller Level Reset and is incremented each time the contents of the log page change (e.g., not only if an Asymmetric Namespace Access Change Asynchronous Event Notification is generated). If the value of this field is FFFFFFFF_FFFFFFFFh, then the field shall be cleared to 0h when incremented (i.e., rolls over to 0h).
09:08	**Number of ANA Group Descriptors:** This field indicates the number of ANA Group Descriptors available in the log page. The log page shall contain one ANA Group Descriptor for each ANA Group that contains namespaces that are attached to the controller. If, for an ANA Group, there are no namespaces attached to the controller processing the command, then no ANA Group Descriptor is returned for that ANA Group (i.e., an ANA Group Descriptor is returned only if that ANA Group contains namespaces that are attached to the controller processing the command. If no namespaces are attached to the controller, then the log page does not contain any ANA Group Descriptors and this field is cleared to 0h.
15:10	Reserved
n:16	**ANA Group Descriptor 0, if any**
m:n+1	**ANA Group Descriptor 1, if any**
…	**…**
x:y	**ANA Group Descriptor n, if any**

The format of the ANA Group Descriptor is defined in Figure 222. Namespace Identifiers shall be listed in ascending NSID order.

Figure 222: ANA Group Descriptor format

Bytes	Description
03:00	**ANA Group ID:** The ANA Group Identifier associated with all namespaces in the ANA Group (refer to section 8.1.2) described by this ANA Group Descriptor.
07:04	**Number of NSID Values:** This field indicates the number of Namespace Identifier values in this ANA Group Descriptor. If the RGO bit is set to ‘1’, then this field is cleared to 0h.
15:08	**Change Count:** This field contains a 64-bit incrementing count, indicating an identifier for the information contained in this ANA Group Descriptor. A value of 0h indicates that the controller does not report a Change Count for this ANA Group Descriptor. If a Change Count is reported, then the count starts at 1h following a Controller Level Reset and is incremented each time the data in this ANA Group Descriptor change. If the value of this field is FFFFFFFF_FFFFFFFFh, then the field shall be set to 1h when incremented (i.e., rolls over to 1h). If this field contains 0h, the host should examine this ANA Group Descriptor for any changes and not use this field as an indicator that a change has occurred.

Figure 222: ANA Group Descriptor format

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>16</td><td>
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>07:04</td><td>Reserved</td></tr>
<tr><td>03:00</td><td>Asymmetric Namespace Access State: This field indicates the Asymmetric Namespace Access state for all namespaces in this ANA Group when accessed through this controller.
<table>
<tr><th>Value</th><th>Description</th><th>Reference</th></tr>
<tr><td>01h</td><td>ANA Optimized state</td><td>8.1.3.1</td></tr>
<tr><td>02h</td><td>ANA Non-Optimized state</td><td>8.1.3.2</td></tr>
<tr><td>03h</td><td>ANA Inaccessible state</td><td>8.1.3.3</td></tr>
<tr><td>04h</td><td>ANA Persistent Loss state</td><td>8.1.3.4</td></tr>
<tr><td>0Fh</td><td>ANA Change state</td><td>8.1.3.5</td></tr>
<tr><td>All other values</td><td>Reserved</td><td></td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>31:17</td><td>Reserved</td></tr>
<tr><td>35:32</td><td>Namespace Identifier 0: The Namespace Identifier of the first namespace that is a member of this ANA Group.</td></tr>
<tr><td>39:36</td><td>Namespace Identifier 1: The Namespace Identifier of the second namespace, if any, that is a member of this ANA Group.</td></tr>
<tr><td>…</td><td>…</td></tr>
<tr><td>((n*4) + 35): ((n*4) + 32)</td><td>Namespace Identifier n: The Namespace Identifier of the n+1 namespace that is a member of this ANA Group.</td></tr>
</table>

5.16.1.14 Persistent Event (Log Identifier 0Dh)

The Persistent Event log page contains information about significant events not specific to a particular command. The information in this log page shall be retained across power cycles and resets. NVM subsystems should be designed for minimal loss of event information upon power failure. This log consists of a header describing the log and zero or more Persistent Events (refer to section 5.16.1.14.1).

This log page is global to the NVM subsystem.

The Log Specific Parameter field in Command Dword 10 (refer to Figure 197) for this log page is defined in Figure 223.

A sanitize operation may alter this log page (e.g., remove or modify events to prevent derivation of user data from log page information, refer to section 8.21). The events removed from this log page by a sanitize operation are unspecified.

Persistent Event Log events specified in this section should be reported in an order such that more recent events are generally reported earlier in the log data than older events. The method by which the NVM subsystem determines the order in which events occurred is vendor specific.

The number of events supported is vendor specific. The supported maximum size for the Persistent Event Log is indicated in the PELS field of the Identify Controller data structure (refer to Figure 275). The number of events supported and the supported maximum size should be large enough that the number of events or the size of the Persistent Event Log data does not reach the maximum supported size over the usable life of the NVM subsystem.

The controller shall log all supported events at each event occurrence unless the controller determines that the same event is occurring at a frequency that exceeds a vendor specific threshold for the frequency of event creation. If the same event is occurring at a frequency that exceeds a vendor specific threshold then the vendor may suppress further entries for the same event. A controller may indicate if events have been suppressed in vendor specific event data.

It is vendor specific which events are deleted (e.g., important events may be retained and events that are newer than an important event that was retained may be deleted) to make room for future events if:

a) the size of the Persistent Event Log data reaches the maximum supported size;
b) the number of events reaches the largest reportable number of events; or
c) an event category reaches the largest reportable number of events for that category (e.g., information regarding 1,000 occurrences of changes to the timestamp is stored in internal data structures and extracted for reporting as Timestamp Change events in the Persistent Event Log and more than 1,000 Timestamp Change events have occurred).

Events that affect multiple controllers (e.g., an NVM Subsystem Reset) should be logged once by a controller selected by the vendor and not logged by any other controllers.

The Action field in the Log Specific Parameter field (refer to Figure 223) specifies whether:

a) A persistent event log reporting context is created at the start of processing this Get Log Page command and log page data, if any, is read from the log page data associated with that log reporting context;
b) Log page data is read from the log page data associated with a preexisting log reporting context; or
c) The persistent event log reporting context, if any, is released.

The persistent event log reporting context is vendor specific information that the controller creates for determining what information is included in the persistent event log page data (e.g., the persistent event log reporting context may be the persistent event log page data or may contain a set of pointers to the events to report).

The controller should retain the persistent event log reporting context:

a) Until the controller processes:

    a) a Get Log Page command requesting the Persistent Event log page with the Action field set to 02h (i.e., Release Context);
    b) an NVM Subsystem Reset; or
    c) a Controller Level Reset;

    or

b) For a vendor specific time long enough to allow retrieval of the Persistent Event log page data.

Persistent Event Log events that occur while a persistent event log reporting context exists shall not be reported in the existing reporting context but shall be logged.

The host is expected to issue a Get Log Page command with the Action field set to 02h to release the persistent event log reporting context after reading the persistent event log page data.

If the Persistent Event Log is not read with a single Get Log Page command, then host software should read the Generation Number field in the Persistent Event Log header after establishing a reporting context but before reading the remainder of the log and then re-read the Generation Number field after it has read the entire log. If the generation numbers do not match, then:

- the reporting context may have been lost while reading the log;
- the Persistent Event Log contents read may be invalid; and
- host software should re-read the log.

Figure 223: Persistent Event Log Specific Parameter Field

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>14:10</td><td>Reserved</td></tr>
<tr><td>09:08</td><td>Action: This field specifies the action the controller shall take during processing this Get Log Page command.

<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>Read Log Data: Return Persistent Event log page data starting at the address indicated by the LPOU field and the LPOL field in the Get Log Page command. If the controller does not have a persistent event log reporting context, then the controller shall abort the command with a status code of Command Sequence Error.</td></tr>
<tr><td>01b</td><td>Establish Context and Read Log Data: The controller shall:

a) determine the length of the Persistent Event log page data;
b) determine the set of events to report in the Persistent Event log page data; and
c) establish a persistent event log reporting context to store information describing the Persistent Event log page data to be reported and track Persistent Event log page data accesses.

After establishing a persistent event log reporting context, the controller shall return Persistent Event log page data starting at the address indicated by the LPOU field and the LPOL field in the Get Log Page command.

If a persistent event log reporting context already exists, then the controller shall abort the command with a status code of Command Sequence Error.</td></tr>
<tr><td>10b</td><td>Release Context: The controller shall:

• release the persistent event log reporting context, if any. It is not an error if the controller does not have a persistent event log reporting context; and
• not return any Persistent Event log page data (i.e., the controller ignores the NUMDU field, NUMDL field, LPOL field, and the LPOU field).</td></tr>
<tr><td>11b</td><td>Establish Context and Read 512 Bytes of Header: The controller shall:

a) determine the length of the Persistent Event log page data;
b) determine the set of events to report in the Persistent Event log page data; and
c) if a reporting context does not already exist, then establish a persistent event log reporting context to store information describing the Persistent Event log page data to be reported and track Persistent Event log page data accesses.

If a persistent event log reporting context did not already exist when the Get Log Page command was processed, then the controller shall:

a) establish a persistent event log reporting context; and

b) after establishing the context, return 512 bytes of the Persistent Event Log Header starting at offset 0h with the Reporting Context Exists bit cleared to ‘0’ with a status code of Successful Completion.

If a persistent event log reporting context already existed when the Get Log Page command was processed, then the controller shall return 512 bytes of the Persistent Event Log Header starting at offset 0h with the Reporting Context Exists bit set to ‘1’ with a status code of Successful Completion.

The 512 bytes of the Persistent Event Log Header shall be returned regardless of the values in the LPOL, LPOU, NUMDL, and NUMDU fields (i.e., the controller shall ignore the LPOL, LPOU, NUMDL, and NUMDU fields).</td></tr>
</table></td></tr>
</table>

The log page returned is defined in Figure 224.

Figure 224: Persistent Event Log Page

Bytes	Description
Persistent Event Log Header	
00	**Log Identifier:** This field shall be set to 0Dh.
03:01	Reserved
07:04	**Total Number of Events (TNEV):** Contains the number of event entries in the log.
15:08	**Total Log Length (TLL):** Contains the total number of bytes of persistent event log page data available, including the header.
16	**Log Revision:** Contains a number indicating the revision of the Get Log Page data structure that this log page data complies with. Shall be set to 03h. This revision applies to the Persistent Event Log and the Persistent Event Format (refer to Figure 225). This revision does not apply to the contents of the Event Data field in the Persistent Event Format as that field is covered by the Event Type Revision (refer to Figure 225).
17	Reserved
19:18	**Log Header Length:** This field contains the length in bytes of the log header information that follows. The total length of the log header in bytes is the value in this field plus 20.
27:20	**Timestamp:** Shall contain a timestamp of the time at which the persistent event log reporting context was established. The value returned shall use the Timestamp data structure defined in Figure 340.
43:28	**Power on Hours (POH):** This field indicates the number of power-on hours at the time the Persistent Event log was retrieved. This may not include time that the controller was powered and in a non-operational state.
51:44	**Power Cycle Count:** Contains the number of power cycles for this controller.
53:52	**PCI Vendor ID (VID):** This is the same value as reported in the Identify Controller data structure PCI Vendor ID field (i.e., bytes 01:00).
55:54	**PCI Subsystem Vendor ID (SSVID):** This is the same value as reported in the Identify Controller data structure PCI Subsystem Vendor ID field (i.e., bytes 03:02).
75:56	**Serial Number (SN):** This field contains the same value as reported in the Serial Number field of the Identify Controller data structure, bytes 23:04.
115:76	**Model Number (MN):** This field contains the same value as reported in the Model Number field of the Identify Controller data structure, bytes 63:24.
371:116	**NVM Subsystem NVMe Qualified Name (SUBNQN):** This field contains the same value as reported in the NVM Subsystem NVMe Qualified Name field of the Identify Controller data structure, bytes 1023:768. If the NVM Subsystem NVMe Qualified Name field of the Identify Controller data structure is not supported, then all bytes of this field shall be cleared to 0h.
373:372	**Generation Number:** Contains a value that is incremented each time a persistent event log reporting context is established and the log page returns different data than when this log page last established a reporting context. If the value of this field is FFFFh, then the field shall be cleared to 0h when incremented (i.e., rolls over to 0h).

Figure 224: Persistent Event Log Page

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>377:374</td><td>
Reporting Context Information (RCI): This field contains information about the persistent event log reporting context.
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>31:19</td><td>Reserved</td></tr>
<tr><td>18</td><td>Reporting Context Exists (RCE): This bit indicates the persistent event log reporting context. If this bit is set to ‘1’, then a persistent event log reporting context already existed when the Get Log Page command that requested this log page was processed. If this bit is cleared to ‘0’, then a persistent event log reporting context did not already exist when the Get Log Page command that requested this log page was processed.</td></tr>
<tr><td>17:16</td><td>Reporting Context Port Identifier Type (RCPIT): If the RCE bit is set to ‘1’, then this field indicates the type of port identifier reported in the Reporting Context Port Identifier (RCPID) field. If the RCE bit is cleared to ‘0’, then this field shall be cleared to 00b.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>A persistent event log reporting context does not already exist.</td></tr>
<tr><td>01b</td><td>The reporting context was established by an NVM subsystem port.</td></tr>
<tr><td>10b</td><td>The reporting context was established by a Management Endpoint (refer to the NVM Express Management Interface Specification).</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>15:00</td><td>Reporting Context Port Identifier (RCPID): If the RCE bit is cleared to ‘0’, then this field shall be cleared to 0h.

If the RCE bit is set to ‘1’, then this field contains a Port Identifier of the type indicated in the RCPIT field.

If the RCPIT field is set to 01b, then this field shall contain the Port Identifier of the NVM subsystem port that established the reporting context as defined in the Primary Controller Capabilities data structure (refer to section 5.17.2.13).

If the RCPIT field is set to 10b, then:
• the least-significant byte of this field shall contain the Port Identifier of the Management Endpoint (refer to the NVM Express Management Interface Specification); and
• the most-significant byte of this field shall be cleared to 0h.</td></tr>
</table>
</td></tr>
<tr><td>479:378</td><td>Reserved</td></tr>
</table>

Figure 224: Persistent Event Log Page

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>511:480</td><td>Supported Events Bitmap: This field contains a bitmap indicating support for the persistent event log events. Each bit in the bitmap corresponds to the value for the event type (refer to Figure 226) (e.g., bit 222 decimal, DEh, corresponds to event type value DEh, Vendor Specific Event). A bit set to ‘1’ indicates that the corresponding event is supported. A bit cleared to ‘0’ indicates that the corresponding event is not supported.
<table>
<tr><th>Bits</th><th>Definition</th><th>Reference</th></tr>
<tr><td>255:224</td><td>Reserved</td><td></td></tr>
<tr><td>223</td><td>TCG Defined</td><td>TCG Storage Interface Interactions Specification</td></tr>
<tr><td>222</td><td>Vendor Specific Event Supported</td><td>5.16.1.14.1.14</td></tr>
<tr><td>221:14</td><td>Reserved</td><td></td></tr>
<tr><td>13</td><td>Thermal Excursion Event Support</td><td>5.16.1.14.1.13</td></tr>
<tr><td>12</td><td>Telemetry Log Create Event Support</td><td>5.16.1.14.1.12</td></tr>
<tr><td>11</td><td>Set Feature Event Support</td><td>5.16.1.14.1.11</td></tr>
<tr><td>10</td><td>Sanitize Completion Event Support</td><td>5.16.1.14.1.10</td></tr>
<tr><td>09</td><td>Sanitize Start Event Support</td><td>5.16.1.14.1.9</td></tr>
<tr><td>08</td><td>Format NVM Completion Event Support</td><td>5.16.1.14.1.8</td></tr>
<tr><td>07</td><td>Format NVM Start Event Support</td><td>5.16.1.14.1.7</td></tr>
<tr><td>06</td><td>Change Namespace Event Support</td><td>5.16.1.14.1.6</td></tr>
<tr><td>05</td><td>NVM Subsystem Hardware Error Event Support</td><td>5.16.1.14.1.5</td></tr>
<tr><td>04</td><td>Power-on or Reset Event Supported</td><td>5.16.1.14.1.4</td></tr>
<tr><td>03</td><td>Timestamp Change Event Supported</td><td>5.16.1.14.1.3</td></tr>
<tr><td>02</td><td>Firmware Commit Event Supported</td><td>5.16.1.14.1.2</td></tr>
<tr><td>01</td><td>SMART / Health Log Snapshot Event Supported</td><td>5.16.1.14.1.1</td></tr>
<tr><td>00</td><td>Reserved</td><td></td></tr>
</table></td></tr>
<tr><th colspan="2">Persistent Event Log Events</th></tr>
<tr><td>(M-1)+512:512</td><td>Persistent Event 0: This field contains the first persistent event log entry (refer to Figure 225) where M is the length of this persistent event.</td></tr>
<tr><td>…</td><td>…</td></tr>
<tr><td>(TLL-1):(TLL-K)</td><td>Persistent Event N: This field contains the last persistent event log entry (refer to Figure 225) where K is the length of this persistent event and TLL is the size specified in the Total Log Length field.</td></tr>
</table>

The format of the Persistent Events in the Persistent Event log is shown in Figure 225.

Figure 225: Persistent Event Format

Bytes	Description
Persistent Event Log Event Header	
00	**Event Type:** This field indicates the event type for this entry. Refer to section 5.16.1.14.1 for the definition of the event types.
01	**Event Type Revision:** This field contains a number indicating the revision of the event data structure for the event indicated by the Event Type field that this event data complies with.
02	**Event Header Length (EHL):** This field contains the length in bytes of the event header information that follows. The total length of the event header in bytes is the value in this field plus 3. The host should use the value in this field to calculate the offset to the start of the Vendor Specific Information field.

Figure 225: Persistent Event Format

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>03</td><td>Event Header Additional Information (EHAI): This field indicates if additional information is present in this event header.
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>7:2</td><td>Reserved</td></tr>
<tr><td>1:0</td><td>Port Identifier Type (PIT): This field indicates the type of port identifier reported in the Port Identifier (PELPID) field. Implementations that are compliant with NVM Express Base Specification revision 1.4 and later shall not clear this field to 0h.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>The Port Identifier Type is not reported and the Port Identifier (PELPID) field does not apply.</td></tr>
<tr><td>01b</td><td>This event is associated with an NVM subsystem port.</td></tr>
<tr><td>10b</td><td>This event is associated with a Management Endpoint (refer to the NVM Express Management Interface Specification).</td></tr>
<tr><td>11b</td><td>This event is not associated with any port.</td></tr>
</table></td></tr>
</table></td></tr>
<tr><td>05:04</td><td>Controller Identifier: This field contains the NVM subsystem unique controller identifier for the controller that created this event. If the event is controller specific, then the event data is associated with that controller. If the event is not controller specific, then this is the controller that the NVM subsystem selected for creating the event.</td></tr>
<tr><td>13:06</td><td>Event Timestamp: This field contains a timestamp of the time when this event occurred using the Timestamp data structure defined in Figure 339.</td></tr>
<tr><td>15:14</td><td>Port Identifier (PELPID): If the PIT field in the EHAI field is cleared to 00b or set to 11b, then this field shall be cleared to 0h.

If the PIT field in the EHAI field is not cleared to 00b or set to 11b, then this field contains a Port Identifier of the type indicated in the PIT field.

If the PIT field in the EHAI field is set to 01b, then this field shall contain the Port Identifier of the NVM subsystem port associated with this event as defined in the Primary Controller Capabilities data structure (refer to section 5.17.2.13).

If the PIT field in the EHAI field is set to 10b, then:
• the least-significant byte of this field shall contain the Port Identifier of the Management Endpoint associated with this event (refer to the NVM Express Management Interface Specification); and
• the most-significant byte of this field shall be cleared to 0h.</td></tr>
<tr><td>19:16</td><td>Reserved</td></tr>
<tr><td>21:20</td><td>Vendor Specific Information Length (VSIL): This field indicates the length in bytes of the Vendor Specific Information. If no Vendor Specific Information is present, then this field shall be cleared to 0h. The length of the Vendor Specific Information is included in the Event Length field (bytes 23:22). Information associated with this event that is not able to be described in the event data structure fields may be reported in Vendor Specific Information fields in this event.</td></tr>
<tr><td>23:22</td><td>Event Length (EL): This field indicates the length in bytes of the vendor specific information, if any, and the persistent event log event data that follows. The total length of the event in bytes is the value in this field plus the value in the Event Header Length field plus 3.</td></tr>
<tr><td colspan="2">Vendor Specific Information, if any</td></tr>
<tr><td>EHL+2+VSIL:EHL+3</td><td>Vendor Specific Information: This field contains the vendor specific information, if any. This field is omitted if there is no vendor specific information (i.e., if VSIL is cleared to 0h).</td></tr>
<tr><td colspan="2">Persistent Event Log Event Data</td></tr>
<tr><td>EHL+EL+2:
EHL+3+VSIL</td><td>Event Data: This field contains persistent event log events (refer to section 5.16.1.14.1).</td></tr>
</table>

5.16.1.14.1Persistent Event Log Events

The values that may be reported in the Event Type field (refer to section 5.16.1.14) of the event header for events in the Persistent Event log are defined in Figure 226.

Figure 226: Persistent Event Log Event Types

Type	O/M[1]	Event	Reference Section
00h		Reserved	
01h	NOTE 2	SMART / Health Log Snapshot	5.16.1.14.1.1
02h	M	Firmware Commit	5.16.1.14.1.2
03h	M	Timestamp Change	5.16.1.14.1.3
04h	M	Power-on or Reset	5.16.1.14.1.4
05h	M	NVM Subsystem Hardware Error	5.16.1.14.1.5
06h	NOTE 3	Change Namespace	5.16.1.14.1.6
07h	NOTE 3	Format NVM Start	5.16.1.14.1.7
08h	NOTE 3	Format NVM Completion	5.16.1.14.1.8
09h	NOTE 3	Sanitize Start	5.16.1.14.1.9
0Ah	NOTE 3	Sanitize Completion	5.16.1.14.1.10
0Bh	O	Set Feature	5.16.1.14.1.11
0Ch	O	Telemetry Log Created	5.16.1.14.1.12
0Dh	O	Thermal Excursion	5.16.1.14.1.13
0Eh to DDh		Reserved	
DEh	O	Vendor Specific Event	5.16.1.14.1.14
DFh	O	TCG Defined	5.16.1.14.1.15
E0h to FFh		Reserved	

Notes:
1. O/M definition: O = Optional, M = Mandatory.
2. Mandatory for NVMe over PCIe implementations, Optional for NVMe over Fabrics implementations.
3. Mandatory if the command used to initiate the activity reported by the event is supported.

5.16.1.14.1.1 SMART / Health Log Snapshot Event (Event Type 01h)

NVM subsystems that support the Persistent Event Log shall create a SMART / Health Log Snapshot Event:

a) If virtualization management is not implemented, then for every controller in the NVM subsystem; or
b) If virtualization management is implemented, then for every primary controller,

at least once every 24 power on hours at a time determined by the controller.

The SMART / Health Log Snapshot Event shall set the Persistent Event Log Event Header:

a) Event Type field to 01h; and
b) Event Type Revision field to 01h.

The SMART / Health Log Snapshot Event data is specified in Figure 227.

Figure 227: SMART / Health Log Snapshot Event Data Format (Event Type 01h)

Bytes	Description
511:00	**Event Data:** Contains a snapshot of the SMART/Health Information Log data specified in Figure 207.

5.16.1.14.1.2 Firmware Commit Event (Event Type 02h)

A firmware commit event shall be recorded in the Persistent Event Log when a Firmware Commit command is completed. The Firmware Commit Event shall set the Persistent Event Log Event Format Header:

a) Event Type field to 02h; and
b) Event Type Revision field to 01h.

The Firmware Commit Event data is specified in Figure 228.

Figure 228: Firmware Commit Event Data Format (Event Type 02h)

Bytes	Description
07:00	**Old Firmware Revision:** Contains the firmware revision of the active firmware before this firmware commit event.
15:08	**New Firmware Revision:** Contains the firmware revision for the firmware that was requested to become active.
16	**Firmware Commit Action:** Contains the value from the Commit Action field in the Firmware Commit command.
17	**Firmware Slot:** Contains the value from the Firmware Slot field in the Firmware Commit command.
18	**Status Code Type for Firmware Commit Command:** Contains the status code type from the completion queue entry for the Firmware Commit command.
19	**Status Returned for Firmware Commit Command:** Contains the status code from the completion queue entry for the Firmware Commit command.
21:20	**Vendor Assigned Firmware Commit Result Code:** Contains a vendor specific value that provides more information about the result of the firmware commit. A value of 0h indicates that no vendor assigned firmware commit result code is provided.

5.16.1.14.1.3 Timestamp Change Event (Event Type 03h)

The Timestamp Change Event (refer to Figure 229) contains the current timestamp, reported in the event header, and the timestamp from the time at which the timestamp was changed (i.e., the old timestamp).

The Timestamp Change Event shall set the Persistent Event Log Event Format Header:

a) Event Type field to 03h; and
b) Event Type Revision field to 01h.

The Timestamp Change Event data is specified in Figure 229.

Figure 229: Timestamp Change Event Format (Event Type 03h)

Bytes	Description
07:00	**Previous Timestamp:** Contains a timestamp of the time immediately before the timestamp was changed (i.e, the old timestamp) using the Timestamp data structure as defined in Figure 339.
15:08	**Milliseconds Since Reset:** Contains the time since the last Controller Level Reset reported in milliseconds.

5.16.1.14.1.4 Power-on or Reset Event (Event Type 04h)

A Power-on or Reset event shall be recorded in the Persistent Event Log when an NVM Subsystem Reset (e.g., due to a power-on) or a Controller Level Reset is completed. The Power-on or Reset Event reports information about resets due to power-on or other events that cause resets (refer to section 3.6) followed by descriptors reporting information about the controller at the time the reset occurred, including timestamp values for all controllers for use in synchronization of timestamp values across controllers.

The controller shall set the Persistent Event Log Event Format Header:

a) Event Type field to 04h; and
b) Event Type Revision field to 01h.

The Power-on or Reset Event data is specified in Figure 230.

Figure 230: Power-on or Reset Event (Event Type 04h)

Bytes	Description
07:00	**Firmware Revision:** Contains the firmware revision that becomes effective when CC.EN transitions from ‘0’ to ‘1’.

Figure 230: Power-on or Reset Event (Event Type 04h)

Bytes	Description
EL-VSIL-1:08[1]	**Reset Information List:** Contains a list of one or more Controller Reset Information descriptors (refer to Figure 231). If virtualization management is not implemented, then the list shall contain a Controller Reset Information descriptor for every controller in the NVM subsystem. If virtualization management is implemented, then the list shall contain a Controller Reset Information descriptor for every primary controller. The Controller Reset Information descriptor is shown in Figure 231.
Notes: 1. Refer to Figure 225 for the definitions of EL and VSIL.	

Figure 231: Controller Reset Information descriptor

Bytes	Description
01:00	**Controller ID:** Contains the Controller ID for the controller with the timestamp in the Controller Timestamp field.
02	**Firmware Activation:** Contains a code indicating if this event triggered a firmware activation. Code / Definition: 00h – Indicates that this event did not trigger a firmware activation on the controller. 01h – Indicates that new firmware was activated on the controller due to this power on or reset. 02h – Indicates that an attempt to activate new firmware on the controller due to this power-on or reset failed. 03h to FFh – Reserved
03	**Operation in Progress:** Bits 7:1 are reserved. Bit 0: A value of '1' indicates that a Format NVM command was in progress for a namespace attached to the controller when this reset event occurred. A value of '0' indicates that no Format NVM commands were in progress for any namespace attached to the controller when this reset event occurred.
15:04	Reserved
19:16	**Controller Power Cycle:** Contains the power cycle count for the controller indicated in the Controller ID field.
27:20	**Power on milliseconds**: Contains the power on hours in milliseconds since being manufactured. This may not include time that the controller was powered and in a non-operational power state. The resolution of this field is vendor specific (e.g., an NVM subsystem that only counts power on time in hours only reports values corresponding to whole hours).
35:28	**Controller Timestamp:** Contains a timestamp for the controller specified in the Controller ID field at the time when this event occurred using the Timestamp data structure defined in Figure 339.

5.16.1.14.1.5 NVM Subsystem Hardware Error Event (Event Type 05h)

An NVM Subsystem Hardware Error event shall be recorded in the Persistent Event Log when a supported NVM subsystem hardware error event is detected. Which of the NVM subsystem hardware error events are supported is vendor specific. The NVM subsystem hardware error event shall set the Persistent Event Log Event Format Header:

- Event Type field to 05h; and
- Event Type Revision Field to 01h.

All detected NVM Subsystem Hardware Error events supported by the NVM subsystem shall be logged unless otherwise specified (e.g., suppressed due to reoccurrence frequency (refer to section 5.16.1.14)).

NVM Subsystem Hardware Error event fields reporting information that is not available (e.g., due to a PCIe optional feature that is not implemented) shall be cleared to 0h unless otherwise specified in the NVM Subsystem Hardware Error Event code description.

The NVM Subsystem Hardware Error Event data is specified in Figure 232.

Figure 232: NVM Subsystem Hardware Error Event Format (Event Type 05h)

Bytes	Description
01:00	**NVM Subsystem Hardware Error Event Code:** This field contains a code (refer to Figure 233) indicating the type of NVM subsystem hardware error that is being reported.
03: 02	Reserved
M+3:04	**Additional Hardware Error Information:** This field contains additional information about the hardware error event indicated in the NVM Subsystem Hardware Error Event Code field (refer to Figure 233). Where M is the number of bytes of additional hardware error information. This field is omitted if the subsystem hardware error being reported does not contain additional hardware error information (i.e., if the number of bytes of additional hardware error information, M, is 0h).

Figure 233: NVM Subsystem Hardware Error Event Codes

Code	Description
00h	Reserved
01h	**PCIe Correctable Error:** Indicates that the NVM subsystem has detected that a PCIe correctable error occurred. Refer to Figure 234 for the format of the Additional Hardware Error Information field.
02h	**PCIe Uncorrectable Non fatal Error:** Indicates that the NVM subsystem has detected that a PCIe uncorrectable non-fatal error occurred. Refer to Figure 234 for the format of the Additional Hardware Error Information field.
03h	**PCIe Uncorrectable Fatal Error:** Indicates that the NVM subsystem has detected that a PCIe uncorrectable fatal error occurred. Refer to Figure 234 for the format of the Additional Hardware Error Information field.
04h	**PCIe Link Status Change:** Indicates that a change in the values reported in the PCI Express Link Status register (refer to the PCI Express Link Status section in the NVMe over PCIe Transport Specification) have changed due to an attempt to correct unreliable link operation. The Additional Hardware Error Information field shall be set to the contents of the PCI Express Link Status register at the time of the event.
05h	**PCIe Link Not Active:** Indicates that the Data Link Control and Management State Machine (refer to PCI Express Base specification) has transitioned out of the DL_Active state without a corresponding event (e.g., without an indication from the host that the link is to be disabled). This NVM subsystem hardware error event does not contain additional hardware error information.
06h	**Critical Warning Condition:** Indicates the NVM subsystem has detected a condition that causes a bit in the Critical Warning field of the SMART / Health Information log (refer to section 5.16.1.3) to be set to '1'. Bits in this field represent the associated state at the time of this event. The Additional Hardware Error Information field shall be set at the time of the event using the same format as is specified for the Critical Warning field of the SMART / Health Information log.

Figure 233: NVM Subsystem Hardware Error Event Codes

<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td>07h</td><td>Endurance Group Critical Warning Condition: Indicates that the NVM subsystem has detected a condition that causes a bit in the Critical Warning field of an Endurance Group Information log page (refer to section 5.16.1.10) to be set to ‘1’.

Bits in this field represent the state at the time this event is added to the Persistent Event log page.

The Additional Hardware Error Information field shall be four bytes long and contain the following information:

<table><tr><th>Bytes</th><th>Definition</th></tr><tr><td>0</td><td>Shall be set at the time this event is added to the Persistent Event log page using the same format as is specified for the Critical Warning field of the Endurance Group Information log page.</td></tr><tr><td>1</td><td>Reserved</td></tr><tr><td>3:2</td><td>Shall be set to the Endurance Group Identifier for the associated Endurance Group.</td></tr></table></td></tr>
<tr><td>08h</td><td>Unsafe Shutdown: Indicates that the controller incremented the Unsafe Shutdowns field value in the SMART / Health Information Log.

The Additional Hardware Error Information field shall be set to the value from the Unsafe Shutdowns field in the SMART / Health Information log at the time of the event.</td></tr>
<tr><td>09h</td><td>Controller Fatal Status: Indicates that the Controller Fatal Status (CSTS.CFS) bit has been set to ‘1’.

This NVM subsystem hardware error event does not contain additional hardware error information.</td></tr>
<tr><td>0Ah</td><td>Media and Data Integrity Status: Indicates that a completion queue entry contained a Media and Data Integrity status code (refer to Figure 98) other than 86h (i.e., Access Denied) or 87h (i.e., Deallocated or Unwritten logical Block).

The Additional Hardware Error Information field shall be set to the contents of the completion queue entry.</td></tr>
<tr><td>0Bh</td><td>Controller Ready Timeout Exceeded: Indicates that:

a) the controller was not ready to process at least one command without error as described in section 3.5.4.1 within the amount of time indicated by:

• the Controller Ready With Media Timeout (CRTO.CRWMT) field in Controller Ready With Media mode (CC.CRIME is cleared to ‘0’); or
• the Controller Ready Independent of Media Timeout (CRTO.CRIMT) field in Controller Ready Independent of Media mode (CC.CRIME is set to ‘1’),

since the controller was enabled; or
b) at least one namespace attached to the controller or media required to process at least one Admin command was not ready within the amount of time indicated by the Controller Ready With Media Timeout (CRTO.CRWMT) field since the controller was enabled by transitioning CC.EN from ‘0’ to ‘1’.

Refer to Figure 235 for the format of the Additional Hardware Error Information field.</td></tr>
<tr><td>0Ch to FFh</td><td>Reserved</td></tr>
</table>

Figure 234: Additional Hardware Error Information for correctable and uncorrectable PCIe errors

Bytes	Value
01:00	**PCIe Device Status Register:** Contains the contents of the PCI Device Status Register (refer to the PCI Express specification) at the time of the event.

Figure 234: Additional Hardware Error Information for correctable and uncorrectable PCIe errors

Bytes	Value
02	Bits 7:1 Reserved Bit 0 **PCIe AER Supported:** set to '1' indicates that PCIe AER (refer to the PCI Express specification) is supported and that the PCIe AER Error Status field, PCIe AER Error Mask field, PCIe AER Header Log Register field, and the PCIe AER TLP Prefix Log Register field are reported. Bit 0 cleared to '0' indicates that PCIe AER is not supported and that the PCIe AER Error Status field, PCIe AER Error Mask field, PCIe AER Header Log Register field, and PCIe AER TLP Prefix Log Register field are not reported (i.e., bytes 79:16 are not reported).
15:03	Reserved
31:16	**PCIe AER Error Status:** Contains the contents of: a) the PCIe AER Correctable Error Status Register (refer to the Advanced Error Reporting Capability section in the NVMe over PCIe Transport Specification) at the time of the event if the error is a correctable error; or b) The PCIe AER Uncorrectable Error Status Register (refer to the Advanced Error Reporting Capability section in the NVMe over PCIe Transport Specification), at the time of the event if the error is an uncorrectable error.
47:32	**PCIe AER Error Mask:** Contains the contents of a) the PCIe AER Correctable Error Mask Register (refer to the Advanced Error Reporting Capability section in the NVMe over PCIe Transport Specification) at the time of the event if the error is a correctable error; or b) the PCIe AER Uncorrectable Error Mask Register (refer to the Advanced Error Reporting Capability section in the NVMe over PCIe Transport Specification) at the time of the event if the error is an uncorrectable error.
63:48	**PCIe AER Header Log Register:** Contains the contents of the PCIe AER Header Log Register (refer to the Advanced Error Reporting Capability section in the NVMe over PCIe Transport Specification), if supported, at the time of the event.
79:64	**PCIe AER TLP Prefix Log Register:** Contains the contents of the PCIe AER TLP Prefix Log Register (refer to the Advanced Error Reporting Capability section in the NVMe over PCIe Transport Specification), if supported, at the time of the event.

Figure 235: Additional Hardware Error Information for Controller Ready Timeout Exceeded errors

<table>
<tr><th>Bytes</th><th>Value</th></tr>
<tr><td>0</td><td>Controller State: Indicates the state of the controller at the time the Controller Ready Timeout Exceeded error occurred.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:4</td><td>Reserved</td></tr>
<tr><td>3</td><td>Controller Not Ready: Indicates the controller was not ready to process at least one command without error as described in section 3.5.4.1 within the amount of time indicated by:
a) the Controller Ready With Media Timeout (CRTO.CRWMT) field in Controller Ready With Media mode (CC.CRIME is cleared to ‘0’); or
b) the Controller Ready Independent of Media Timeout (CRTO.CRIMT) fied in Controller Ready Independent of Media mode (CC.CRIME is set to ‘1’),
since the controller was enabled.</td></tr>
<tr><td>2</td><td>Admin Command Media Not Ready: Indicates media required to process at least one Admin command was not ready within the amount of time indicated by the Controller Ready With Media Timeout (CRTO.CRWMT) field since the controller was enabled by transitioning CC.EN from ‘0’ to ‘1’.</td></tr>
<tr><td>1</td><td>Namespace Not Ready: Indicates at least one namespace attached to the controller was not ready within the amount of time indicated by the Controller Ready With Media Timeout (CRTO.CRWMT) field since the controller was enabled by transitioning CC.EN from ‘0’ to ‘1’.</td></tr>
<tr><td>0</td><td>Controller Ready Independent of Media Enable: Indicates the value of the CC.CRIME bit when the Controller Ready Timeout Exceeded error occurred.</td></tr>
</table></td></tr>
<tr><td>3:1</td><td>Reserved</td></tr>
</table>

5.16.1.14.1.6 Change Namespace Event (Event Type 06h)

The Changed Namespace Event (refer to Figure 236) persists the host parameters used for successful Namespace Management commands. The event contains a Persistent Event Log Event header and the Change Namespace Event data.

The Changed Namespace Event shall set the Persistent Event Log Event Format Header:

- Event Type field to 06h; and
- Event Type Revision Field to 02h.

Figure 236: Change Namespace Event Data Format (Event Type 06h)

Bytes	Value
03:00	**Namespace Management CDW10:** Contains the value from command Dword 10 of the Namespace Management command that initiated the namespace change event (refer to Figure 298).
07:04	Reserved
15:08	**Namespace Size (NSZE):** For a create operation, contains the NSZE value from the Identify Namespace data structure in the Namespace Management command (refer to Figure 300). For a delete operation that specifies a single namespace this field contains the value from the NSZE field of the Identify Namespace data structure (refer to the I/O Command Set specific Identify Namespace data structure in the applicable I/O Command Set specification) for the namespace being deleted. For a delete operation that specifies all namespaces this field is reserved.
23:16	Reserved

Figure 236: Change Namespace Event Data Format (Event Type 06h)

Bytes	Value
31:24	**Namespace Capacity (NCAP):** For a creation operation, contains the NCAP value from the Identify Namespace data structure in the Namespace Management command (refer to Figure 300). For a delete operation that specifies a single namespace this field contains the value from the NCAP field of the Identify Namespace data structure (refer to the I/O Command Set specific Identify Namespace data structure in the applicable I/O Command Set specification) for the namespace being deleted. For a delete operation that specifies all namespaces this field is reserved. For I/O Command Sets that don't define this field, it is considered reserved.
32	**Formatted LBA Size (FLBAS):** Refer to the applicable I/O Command Set specification for details. For I/O Command Sets that don't define this field, it is considered reserved. NOTE: This field applies to all User Data Formats. The original name has been retained for historical continuity.
33	**End-to-end Data Protection Type Settings (DPS):** Refer to the applicable I/O Command Set specification for details. For I/O Command Sets that don't define this field, it is considered reserved.
34	**Namespace Multi-path I/O and Namespace Sharing Capabilities (NMIC):** For a create operation, contains the NMIC value from the Namespace Management – Host Software Specified Fields data structure defined in the applicable I/O Command Set specification. For a delete operation that specifies a single namespace this field contains the value from the NMIC field of the I/O Command Set Independent Identify Namespace data (refer to Figure 280[1]) for the namespace being deleted. For a delete operation that specifies all namespaces this field is reserved.
35	Reserved
39:36	**ANA Group Identifier (ANAGRPID):** For a create operation, contains the ANAGRPID value from the Namespace Management – Host Software Specified Fields data structure defined in the applicable I/O Command Set specification, if specified, or contains the ANAGRPID value from the I/O Command Set Independent Identify Namespace data (refer to Figure 280[1]) after the namespace was created if an ANA Group Identifier was not specified in the command. For a delete operation that specifies a single namespace this field contains the value from the ANAGRPID field of the I/O Command Set Independent Identify Namespace data (refer to Figure 280[1]) for the namespace being deleted. For a delete operation that specifies all namespaces this field is reserved. If ANA Groups are not supported, then the ANAGRPID field shall be cleared to 0h.
41:40	**NVM Set Identifier (NVMSETID):** For a create operation, contains the NVMSETID value from the Namespace Management – Host Software Specified Fields data structure defined in the applicable I/O Command Set specification. For a delete operation that specifies a single namespace this field contains the value from the NVMSETID field of the Identify Namespace data (refer to Figure 280[1]) for the namespace being deleted. For a delete operation that specifies all namespaces this field is reserved.
43:42	**Endurance Group Identifier (ENDGID):** For a create operation, contains the ENDGID value from the Namespace Management – Host Software Specified Fields data structure defined in the applicable I/O Command Set specification. For a delete operation that specifies a single namespace, this field contains the value from the ENDGID field of the Identify Namespace data structure (refer to Figure 280[1]) for the namespace being deleted. For a delete operation that specifies all namespaces this field is reserved.
47:44	**Namespace ID (NSID):** For a create operation, contains the NSID for the namespace that was created. For a delete operation, contains the NSID for the namespace being deleted or FFFFFFFFh for a delete operation specifying all namespaces.

Notes:
1. For controllers that implement the NVM Command Set, this field contains the value of the associated field from the Identify Namespace data structure (refer to the NVM Command Set Identify Namespace data structure in the NVM Command Set Specification).

5.16.1.14.1.7 Format NVM Start Event (Event Type 07h)

A Format NVM Start event shall be recorded in the Persistent Event Log after successfully validating the command parameters of a Format NVM Command (refer to section 5.14) and before modifying any of the contents of the NVM.

The Format NVM Start event shall set the Persistent Event Log Event Format Header:

- Event Type field to 07h; and
- Event Type Revision field to 01h.

Figure 237: Format NVM Start Event Data Format (Event Type 07h)

Bytes	Description
03:00	**Namespace Identifier:** Contains the namespace identifier specified in the Format NVM command.
04	**Format NVM Attributes (FNA):** Contains the value from the identify controller FNA field.
07:05	Reserved
11:08	**Format NVM CDW10:** Contains the value from command Dword 10 of the Format NVM command (refer to Figure 189).

5.16.1.14.1.8 Format NVM Completion Event (Event Type 08h)

A Format NVM Completion event shall be recorded in the Persistent Event Log at the completion of a Format NVM command that resulted in modification of the contents of the NVM.

The Format NVM Completion event shall set the Persistent Event Log Event Format Header:

- Event Type field to 08h; and
- Event Type Revision field to 02h.

Figure 238: Format NVM Completion Event Data Format (Event Type 08h)

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>03:00</td><td>Namespace Identifier: Contains the namespace identifier specified in the Format NVM command.</td></tr>
<tr><td>04</td><td>Smallest Format Progress Indicator: For a Format NVM command that formats a single namespace, this field contains the lowest numerical value that was available for reporting in the FPI field of the Identify Namespace data structure (i.e., if the format did not complete successfully and the FPI field is supported, then this field contains the percentage of the namespace that remained to be formatted at the time the Format NVM command completed, refer to Figure 280) during the format operation. For a Format NVM command that formats all namespaces this field shall be cleared to 0h.</td></tr>
<tr><td>05</td><td>Format NVM Status: Contains the status of the format operation.
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>7:2</td><td>Reserved</td></tr>
<tr><td>1</td><td>Incomplete Format: If set to ‘1’, then the format operation modified some or all of the user data but did not complete successfully. If set to ‘1’, then the Format NVM Error bit shall be set to ‘1’. If cleared to ‘0’, then the format operation either did not modify any user data or the format operation completed successfully.</td></tr>
<tr><td>0</td><td>Format NVM Error: If set to ‘1’, then the format operation did not complete successfully. If cleared to ‘0’, then the format operation completed successfully.</td></tr>
</table></td></tr>
<tr><td>07:06</td><td>Completion Information: Contains a vendor specific value that may provide more information about the completion of the format operation (e.g., if the format operation did not complete successfully, then this field may contain a vendor specific code that indicates a vendor specific reason).</td></tr>
</table>

Figure 238: Format NVM Completion Event Data Format (Event Type 08h)

Bytes	Description
09:08	<table><tr><th>Bits</th><th>Description</th></tr><tr><td>15:1</td><td>**Status:** This field indicates the value that was reported in the Status field for the completion queue entry, if any, for the Format NVM command associated with this event. If no completion queue entry was reported, then this field shall be cleared to 0h.</td></tr><tr><td>0</td><td>**Phase Tag:** This field may indicate the Phase Tag posted for the command.</td></tr></table>
11:10	Reserved

5.16.1.14.1.9 Sanitize Start Event (Event Type 09h)

A Sanitize Start event shall be recorded in the Persistent Event Log at the start of a sanitize operation.

The Sanitize Start event shall set the Persistent Event Log Event Format Header:

- Event Type field to 09h; and
- Event Type Revision field to 01h.

Figure 239: Sanitize Start Event Data Format (Event Type 09h)

Bytes	Description
03:00	**SANICAP:** Contains the contents of the SANICAP field from the Identify Controller data structure.
07:04	**Sanitize CDW10:** Contains the value from command Dword 10 of the Sanitize command (refer to Figure 303).
11:08	**Sanitize CDW11:** Contains the value from command Dword 11 of the Sanitize command (refer to Figure 304).

5.16.1.14.1.10 Sanitize Completion Event (Event Type 0Ah)

A Sanitize Completion event shall be recorded in the Persistent Event Log at the completion of a sanitize operation.

The Sanitize Completion event shall set the Persistent Event Log Event Format Header:

- Event Type field to 0Ah; and
- Event Type Revision field to 01h.

Figure 240: Sanitize Completion Event Data Format (Event Type 0Ah)

Bytes	Description
1:0	**Sanitize Progress:** Contains the sanitize progress at the time of this event using the format specified for the SPROG field in the Sanitize Status log page (refer to section 5.16.1.25).
3:2	**Sanitize Status:** Contains the sanitize status for the time of this event using the format specified for the SSTAT field in the Sanitize Status log page. (e.g., the Global Data Erase bit indicates the status at the time of this event).
5:4	**Completion Information:** Contains a vendor specific value that may provide more information about the completion of the sanitize operation (e.g., if the sanitize operation did not complete successfully, then this field may contain a vendor specific code that indicates a vendor specific reason).
7:6	Reserved

5.16.1.14.1.11 Set Feature Event (Event Type 0Bh)

The Set Feature Event persists the data of a successful Set Features command. The event contains a Persistent Event Log Event header and the Set Feature Event data (refer to Figure 241).

The Set Feature Event shall set the Persistent Event Log Event Format Header:

- Event Type field to 0Bh; and

- Event Type Revision Field to 01h.

A Set Feature Event shall be recorded in the Persistent Event Log when the following criteria are met:

a) A Set Features command completes successfully;
b) The Feature Identifier in that Set Features command is supported to be logged in the Persistent Event Log; and
c) There is a change to the controller settings for the Feature Identifier in that Set Features command (i.e., the same setting is not set again).

A Set Feature Event may be recorded in the Persistent Event Log when there is no change to the controller settings for the Feature Identifier in that Set Features command if the following criteria are met:

a) A Set Features command completes successfully; and
b) The Feature Identifier in that Set Features command is supported to be logged in the Persistent Event Log.

The Feature Identifiers that may be supported to be logged in the Persistent Event Log are shown in Figure 25, Figure 30, and Figure 34.

The Command Dwords and Memory Buffer logged in the Set Feature Event data use the same formats as the formats defined by the Set Features and Get Features commands.

Figure 241: Set Feature Event Data Format

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>03:00</td><td>Set Feature Event Layout: Defines the number of Command Dwords and the amount of data in the Memory Buffer from the Set Features command associated with this event.
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>31:16</td><td>Memory Buffer Count: Defines the number of bytes from the memory buffer that are logged in the Memory Buffer field. A value of 0h indicates that the Memory Buffer field does not exist.</td></tr>
<tr><td>15:04</td><td>Reserved</td></tr>
<tr><td>03</td><td>Logged Command Completion Dword 0: If set to ‘1’, then Dword 0 of the command completion for the Set Features command is included in the log. If cleared to ‘0’, then Dword 0 of the command completion command for the Set Features command is not included in the log.</td></tr>
<tr><td>02:00</td><td>Dword Count: contains the number of consecutive Dwords starting with Dword 10 from the Set Features command that are reported in the Command Dwords field. The values 0h and 7h are reserved.</td></tr>
</table></td></tr>
<tr><td>(Dword Count * 4)+3: 4</td><td>Command Dwords: Contains a sequential list of Command Dwords from the Set Features command starting with Command Dword 10. The number of entries in the list is specified by the Command Dword Count field. All non-reserved Command Dwords specified by the Set Features command for the Feature Identifier shall be logged. The Command Dwords are ordered as defined by the Common Command Format in Figure 87.</td></tr>
<tr><td>(Memory Buffer Count) + (Dword Count * 4) + 3: (Dword Count * 4) + 4</td><td>Memory Buffer: Contains the data in the memory buffer for the Set Features command.

If the Memory Buffer Count field is cleared to 0h, then this field does not exist in the logged event.</td></tr>
<tr><td>Memory Buffer Count + (Dword Count * 4)+7: Memory Buffer Count + (Dword Count * 4)+4</td><td>Command Completion Dword 0: If the Logged Command Completion Dword 0 bit is set to ‘1’, then this field contains the Dword 0 value from the Set Features command completion. If the Logged Command Completion Dword 0 bit is cleared to ‘0’, then this field is not logged.</td></tr>
</table>

5.16.1.14.1.12 Telemetry Log Create Event (Event Type 0Ch)

A Telemetry Log Create event may be created if the controller determines that a Telemetry Host-Initiated log page (refer to section 5.16.1.8) or that a Telemetry Controller-Initiated log page (refer to section 5.16.1.9) has been generated.

The Telemetry Log Create Event shall set the Persistent Event Log Event Format Header:

- Event Type field to 0Ch; and
- Event Type Revision Field to 01h.

Figure 242: Telemetry Log Create Event Data Format (Event Type 0Ch)

Bytes	Description
511:00	**Telemetry Initiated Log:** Contains a copy of the values from the first 512 bytes of the Telemetry Host-Initiated log page (refer to Figure 215) or the first 512 bytes of the Telemetry Controller-Initiated log page (refer to Figure 216).

5.16.1.14.1.13 Thermal Excursion Event (Event Type 0Dh)

A Thermal Excursion event shall be recorded in the Persistent Event Log if the Composite Temperature has transitioned from a temperature that is less than:

a) the WCTEMP, if any, (refer to Figure 275) to a temperature that is greater than or equal to the WCTEMP, if any; or
b) the CCTEMP, if any, (refer to Figure 275), to a temperature that is greater than or equal to the CCTEMP, if any,

unless recording of the event causes a vendor specific frequency of threshold reports for this threshold to be exceeded.

A Thermal Excursion event may be recorded in the Persistent Event Log if the Composite Temperature has transitioned from a temperature:

a) that is less than TMT1 (refer to section 5.27.1.13), if any, to a temperature that is greater than or equal to TMT1, if any (i.e., light throttling has started);
b) that is less than TMT2 (refer to section 5.27.1.13), if any, to a temperature that is greater than or equal to TMT2, if any (i.e., heavy throttling has started);
c) that is less than a vendor specific temperature where thermal throttling occurs due to self-throttling to a temperature that is greater than that vendor specific temperature (i.e., self-throttling has started);
d) outside of a temperature threshold to a value that is within all temperature thresholds (i.e., the temperature returns to normal);
e) at which thermal throttling is occurring to a temperature at which thermal throttling is stopped; or
f) that is greater than an under temperature threshold (refer to section 5.27.1.3) to a temperature that is less than or equal to an under temperature threshold,

unless recording of the event causes a vendor specific frequency of threshold reports for this threshold to be exceeded.

The Thermal Excursion event shall set the Persistent Event Log Event Format Header;

- Event Type field to 0Dh; and
- Event Type Revision field to 01h.

Figure 243: Thermal Excursion Event Data Format (Event Type 0Dh)

Bytes	Description
0	**Over Temperature:** Contains the absolute value of the difference (i.e., delta) in Kelvins between the temperature indicated in the threshold field and temperature measured at the time of the event.

Figure 243: Thermal Excursion Event Data Format (Event Type 0Dh)

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>1</td><td>Threshold: Contains an indicator for the temperature threshold crossing that is being reported.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><th colspan="2">High Temperature Transitions</th></tr>
<tr><td>1h</td><td>The Composite Temperature has transitioned from a temperature that is less than WCTEMP to a temperature that is greater than or equal to WCTEMP.</td></tr>
<tr><td>2h</td><td>The Composite Temperature has transitioned from a temperature that is less than CCTEMP to a temperature that is greater than or equal to CCTEMP.</td></tr>
<tr><td>3h</td><td>The Composite Temperature has transitioned from a temperature that is less than TMT1 to a temperature is greater than or equal to TMT1 (i.e., vendor specific thermal management actions that minimize the impact on performance, such as light throttling, have started).</td></tr>
<tr><td>4h</td><td>The Composite Temperature has transitioned from a temperature that is less than TMT2 to a temperature that is greater than or equal to TMT2 (i.e., vendor specific thermal management actions that may impact performance, such as heavy throttling, have started).</td></tr>
<tr><td>5h</td><td>The Composite Temperature has transitioned from a temperature where no vendor specific thermal management actions are taken to a temperature that is greater than or equal to a vendor specific temperature at which vendor specific thermal management actions have started (e.g., self-throttling).</td></tr>
<tr><th colspan="2">Normal Temperature Transitions</th></tr>
<tr><td>88h</td><td>The Composite Temperature has transitioned from a temperature that is greater than or equal to WCTEMP or is less than or equal to an under temperature threshold to a temperature that is between WCTEMP and that under temperature threshold (i.e., the temperature has transitioned to a normal temperature).</td></tr>
<tr><td>89h</td><td>The Composite Temperature has transitioned from a temperature that is greater than a temperature where thermal management actions are being performed and that is not greater than or equal to WCTEMP to a temperature where thermal management actions are stopped.</td></tr>
<tr><th colspan="2">Low Temperature Transitions</th></tr>
<tr><td>B0h</td><td>The Composite Temperature has transition from a temperature that is greater than an under temperature threshold to a temperature that is less than or equal to an under temperature threshold.</td></tr>
<tr><td>F0h to FFh</td><td>Vendor specific</td></tr>
<tr><td>All other values</td><td>Reserved</td></tr>
</table></td></tr>
</table>

5.16.1.14.1.14 Vendor Specific Event (Event Type DEh)

The Vendor Specific Event (refer to Figure 244) contains a set of Vendor Specific Event Descriptors that describe an event that the vendor has determined is a significant event which should be reported to a host in the persistent event log and that is not described by any of the other persistent event log events.

The Vendor Specific Event Descriptors follow the format shown in Figure 245 and contain vendor specific data of the type indicated in the Vendor Specific Event Data Type field of the Vendor Specific Event Descriptor.

If a UUID Index is specified in the Get Log Page command (refer to section 5.16), then the controller shall return:

a) Vendor specific events defined by the vendor identified by the specified UUID index; and
b) Vendor specific events defined by the NVM subsystem manufacturer.

The controller shall set the Vendor Specific Event Format Header:

a) Event Type field to DEh; and
b) Event Type Revision field to 01h.

The Vendor Specific Event data is specified in Figure 244.

Figure 244: Vendor Specific Event Format (Event Type DEh)

Bytes	Description
M-1:0	**Vendor Specific Event Descriptor 0:** Contains the first vendor specific event descriptor (refer to Figure 245). Where M is the length of this vendor specific event descriptor.
…	
EL-VSIL-1: EL-VSIL-K [1]	**Vendor Specific Event Descriptor N:** Contains the last vendor specific event descriptor (refer to Figure 245). Where K is the length of this vendor specific event descriptor (refer to Figure 245).

Notes:
1. Refer to Figure 225 for the definitions of EL and VSIL.

The format of the Vendor Specific Event Descriptor is shown in Figure 245.

Figure 245: Vendor Specific Event Descriptor Format

Bytes	Description
01:00	**Vendor Specific Event Code:** Contains a vendor specific code that uniquely identifies the type of event that is described in the data that follows. All vendor specific events of the same type should report the same Vendor Specific Event Code field value.
02	**Vendor Specific Event Data Type:** Contains a code indicating the type of data reported in the Vendor Specific Event Data field (refer to Figure 246).
03	**UUID Index:** UUID Index (refer to Figure 477) at the time of this event for the vendor that defined this event.
05:04	**Vendor Specific Event Data Length (VSEDL)**: Contains the length in bytes of the Vendor Specific Event Data field.
Vendor Specific Event Data, if any (i.e., VSEDL > 0)	
VSEDL+5:06	**Vendor Specific Event Data:** Contains vendor specific data that is associated with this event and is of the type specified in the Vendor Specific Event Data Type field.

The Vendor Specific Event Data Types that are able to be reported in a Vendor Specific Event Descriptor are shown in Figure 246.

Figure 246: Vendor Specific Event Data Type Codes

Code	Description
00h	Reserved
01h	**Event Name:** The Vendor Specific Event Data field contains a null terminated ASCII string with a vendor specific name for the value in the Vendor Specific Event Code field. The value reported in this field shall be the same for every vendor specific event containing a vendor specific event code that is the same as the value in the Vendor Specific Event Code field in this event. If a Vendor Specific Event Descriptor specifying this data type is reported, then that descriptor shall be the first Vendor Specific Event Descriptor in that event.
02h	**ASCII String:** The Vendor Specific Event Data field contains a null terminated ASCII string.
03h	**Binary:** The Vendor Specific Event Data field contains binary data. The byte ordering in the binary data is determined by the NVM subsystem vendor.
04h	**Signed Integer:** The Vendor Specific Event Data field contains a 64-bit signed integer in two's complement form.
05h to FFh	Reserved

5.16.1.14.1.15 TCG Defined Event (Event Type DFh)

The TCG Defined Event shall set the Persistent Event Log Event Format Header:

- Event Type field to DFh.

The Event Type Revision Field and the TCG Defined Event data are reserved for TCG.

5.16.1.15 Endurance Group Event Aggregate (Log Identifier 0Fh)

This log page indicates if an Endurance Group Event (refer to section 3.2.3) has occurred for a particular Endurance Group. If an Endurance Group Event has occurred, the details of the particular event are included in the Endurance Group Information log page for that Endurance Group. An asynchronous event is generated when an entry for an Endurance Group is newly added to this log page.

If there is an enabled Endurance Group Event pending for an Endurance Group, then the Endurance Group Event Aggregate log page includes an entry for that Endurance Group. The log page is an ordered list by Endurance Group Identifier. For example, if Endurance Group Events are pending for Endurance Group 2, 1, and 7, then the log page shall have entries in numerical order of 1, 2, and 7. A particular Endurance Group entry is removed from this log page after the Get Log Page command is completed successfully with the Retain Asynchronous Event bit cleared to '0' for the Endurance Group Information log page for that Endurance Group.

The log page size is limited by the Endurance Group Identifier Maximum value reported in the Identify Controller data structure (refer to Figure 275). If the host reads beyond the end of the log page, zeroes are returned. The log page is defined in Figure 247.

Figure 247: Endurance Group Event Aggregate Log Page

Bytes	Description
07:00	**Number of Entries:** This field indicates the number of entries in the list. The maximum number of entries in the list corresponds to the Endurance Group Identifier Maximum field reported in the Identify Controller data structure. A value of 0h indicates there are no entries in the list.
09:08	**Entry 1:** Indicates the Endurance Group that has an Endurance Group Event pending that has the numerically smallest Endurance Group Identifier, if any.
11:10	**Entry 2:** Indicates the Endurance Group that has an Endurance Group Event pending that has the second numerically smallest Endurance Group Identifier, if any.
13:12	**Entry 3:** Indicates the Endurance Group that has an Endurance Group Event pending that has the third numerically smallest Endurance Group Identifier, if any.
15:14	**Entry 4:** Indicates the Endurance Group that has an Endurance Group Event pending that has the fourth numerically smallest Endurance Group Identifier, if any.
…	…
2*n+7:2*n+6	**Entry n:** Indicates the Endurance Group that has an Endurance Group Event pending that has the numerically largest Endurance Group Identifier, if any.

5.16.1.16 Media Unit Status (Log Identifier 10h)

This log page is used to describe the configuration and wear of Media Units (refer to section 8.3). The log page contains one Media Unit Status Descriptor for each Media Unit accessible by the specified domain. Each Media Unit Status Descriptor (refer to Figure 249) indicates the configuration of the Media Unit (e.g., to which Endurance Group the Media Unit is assigned, to which NVM Set the Media Unit is assigned, to which Channels the Media Unit is attached) and indications of wear (e.g., the Available Spare field and the Percentage Used field). The indications of wear change as the Media Unit is written and read.

If the NVM subsystem supports multiple domains, then the controller reports the Media Unit Status log page for the domain specified in the Log Specific Identifier field (refer to Figure 198), if accessible. If the information is not accessible, then the log page is not available (refer to section 8.1.4). If the Log Specific Identifier field is cleared to 0h, then the specified domain is the domain containing the controller that is processing the command.

Media Unit Identifier values (refer to Figure 249) begin with 0h and increase sequentially. If the NVM subsystem supports multiple domains, then the Media Unit Identifier values are unique within the specified domain. If the NVM subsystem does not support multiple domains, then the Media Unit Identifier values are unique within the NVM subsystem.

Media Unit Status Descriptors are listed in ascending order by Media Unit Identifier.

Requirements for supporting the Media Unit Status log page are defined in section 8.3.

Figure 248: Media Unit Status Log Page

Bytes	Description
01:00	**Number of Media Unit Status Descriptors (NMU):** This field indicates the number of Media Unit Status Descriptors in the log page. If this field is cleared to 0h, then no Media Unit Status Descriptors are reported.
03:02	**Number of Channels (CCHANS):** This field indicates the number of Channels accessible by the controller. A value of 0h indicates that the number of Channels accessible by the controller is not reported.
05:04	**Selected Configuration:** This field indicates the Configuration Identifier selected by the most recent successful completion of the Capacity Management command Select Capacity Configuration operation. If a Select Capacity Configuration operation has not been completed, this field may indicate a non-zero value (i.e., a configuration was selected by default). If a Configuration Identifier is not selected, then this field shall be cleared to 0h.
15:6	Reserved
NOTE 1	**Media Unit Status Descriptor 0:** This field contains the first Media Unit Status Descriptor (refer to Figure 249), if any.
NOTE 1	**Media Unit Status Descriptor 1:** This field contains the second Media Unit Status Descriptor, if any.
	...
NOTE 1	**Media Unit Status Descriptor NMU-1:** This field contains the last Media Unit Status Descriptor, if any.
Notes: 1. Media Unit Status Descriptors may be different lengths.	

The Media Unit Status Descriptor is defined in Figure 249.

If the Selected Configuration field is cleared to 0h, then the following fields in each Media Unit Status Descriptor shall be cleared to 0h:

a) Endurance Group Identifier (ENDGID);
b) NVM Set Identifier (NVMSETID);
c) Capacity Adjustment Factor; and
d) Number of Channels (MUCS).

Channel Identifier values begin with 0h and increase sequentially. If the NVM subsystem supports multiple domains, then Channel Identifier values are unique within the specified domain. If the NVM subsystem does not support multiple domains, then Channel Identifier values are unique within the NVM subsystem.

In the Media Unit Status Descriptor, Channel Identifiers are listed in ascending order by value, and each Channel Identifier shall appear only once.

Figure 249: Media Unit Status Descriptor

Bytes	Description
01:00	**Media Unit Identifier:** This field indicates the identifier of the Media Unit.
03:02	**Domain Identifier:** This field indicates the identifier of the domain that contains this Media Unit. Refer to section 3.2.4.3. A value of 0h indicates that a Domain Identifier is not reported. If multiple domains are not supported, then this field shall be cleared to 0h.
05:04	**Endurance Group Identifier (ENDGID):** This field indicates the identifier of the Endurance Group that contains this Media Unit. Refer to section 3.2.3. The value shall be less than or equal to the value of the Endurance Group Identifier Maximum field (refer to Figure 275). A value of 0h indicates that this Media Unit is not part of an Endurance Group.
07:06	**NVM Set Identifier (NVMSETID):** This field indicates the identifier of the NVM Set that contains this Media Unit. Refer to section 3.2.2. This field shall indicate a value less than or equal to the value of the NVM Set Identifier Maximum field (refer to Figure 275). If the controller does not support NVM Sets, then this field shall be cleared to 0h.

Figure 249: Media Unit Status Descriptor

Bytes	Description
09:08	**Capacity Adjustment Factor:** This field indicates the capacity adjustment factor (refer to section 8.3.1) for this Endurance Group. A value of FFFFh indicates that value and all higher values. A value of 0h indicates that the Capacity Adjustment Factor is not reported. All Media Unit Status Descriptors which indicate the same Endurance Group Identifier shall indicate the same value in their Capacity Adjustment Factor fields.
10	**Available Spare:** Contains a normalized percentage (0 to 100%) of the remaining spare capacity available for the Media Unit. The relationship between this value and the value in the Available Spare field in the Endurance Group Information log page (refer to section 5.14.1.9) is outside the scope of this specification.
11	**Percentage Used:** Contains a vendor specific estimate of the percentage of life used for the Media Unit based on the actual usage and the manufacturer's prediction of NVM life. A value of 100 indicates that the estimated endurance of the NVM in the Media Unit has been consumed, but may not indicate an NVM failure. The value is allowed to exceed 100. Percentages greater than 254 shall be represented as 255. This value shall be updated once per power-on hour when the controller is not in a sleep state. Refer to the JEDEC JESD218A standard for SSD device life and endurance measurement techniques. The relationship between this value and the value in the Percentage Used field in the Endurance Group Information log page is outside the scope of this specification.
12	**Number of Channels (MUCS):** This field indicates the number of Channels attached to this Media Unit. If this field is cleared to 0h, then no Channel Identifiers are reported for this Media Unit.
13	**Channel Identifiers Offset (CIO):** This field indicates the offset of the Channel 0 Identifier field from the beginning of the Media Unit Status Descriptor. This field shall be a non-zero value and a multiple of 16.
CIO-1:14	Reserved
Channel Identifiers	
CIO+1 : CIO	**Channel 0 Identifier:** This field contains the identifier for the first Channel attached to this Media Unit, if any.
CIO+3 : CIO+2	**Channel 1 Identifier:** This field contains the identifier for the second Channel attached to this Media Unit, if reported, if any.
	…
(MUCS*2)+CIO-1 : (MUCS*2)+CIO-2	**Channel MUCS-1 Identifier:** This field contains the identifier for the last Channel attached to this Media Unit, if any.

5.16.1.17 Supported Capacity Configuration List (Log Identifier 11h)

This log page is used to provide a list of Supported Capacity Configuration Descriptors (refer to Figure 250). Each entry in the list defines a different configuration of Endurance Groups supported by the specified domain.

If the NVM subsystem supports multiple domains, then the controller reports the Supported Capacity Configuration List log page for the domain specified in the Log Specific Identifier field (refer to Figure 198), if accessible. If the information is not accessible, then the log page is not available (refer to section 8.1.3). If the Log Specific Identifier field is cleared to 0h, then the specified domain is the domain containing the controller that is processing the command.

If the NVM subsystem supports multiple domains, then Capacity Configuration Identifier values are unique within the specified domain. If the NVM subsystem does not support multiple domains, then Capacity Configuration Identifier values are unique within the NVM subsystem.

In the Supported Capacity Configuration List (refer to Figure 250), Capacity Configuration Descriptors shall be listed in ascending order by Capacity Configuration Identifier, and each Capacity Configuration Identifier shall appear only once.

Figure 250: Supported Capacity Configuration List Log Page

Bytes	Description
0	**Number of Supported Capacity Configurations (SCCN):** This field indicates the number of Capacity Configuration Descriptors in the list. If this field is cleared to 0h, then no Capacity Configuration Descriptors are reported.
15:1	Reserved
NOTE 1	**Capacity Configuration 0 Descriptor:** This field indicates the first Capacity Configuration Descriptor (refer to Figure 251) in the list, if any.
NOTE 1	**Capacity Configuration 1 Descriptor:** This field indicates the second Capacity Configuration Descriptor in the list, if any.
…	
NOTE 1	**Capacity Configuration SCCN-1 Descriptor:** This field indicates the last Capacity Configuration Descriptor in the list, if any.
Notes: 1. Capacity Configuration Descriptors may be different lengths.	

The Capacity Configuration Descriptor (refer to Figure 251) indicates the details of a particular configuration of Endurance Groups and contains one Endurance Group Configuration Descriptor for each Endurance Group accessible by the controller processing the command.

In the Capacity Configuration Descriptor (refer to Figure 251), Endurance Group Configuration Descriptors shall be listed in ascending order by Endurance Group Identifier, and each Endurance Group Identifier shall appear only once.

Figure 251: Capacity Configuration Descriptor

Bytes	Description
1:0	**Capacity Configuration Identifier:** This field indicates the identifier for this Capacity Configuration.
3:2	**Domain Identifier:** This field indicates the identifier of the domain (refer to section 3.2.4.3) containing the Endurance Group configurations described by this Capacity Configuration Descriptor. A value of 0h indicates that a Domain Identifier is not reported. If multiple domains are not supported, then this field shall be cleared to 0h.
5:4	**Number of Endurance Group Configuration Descriptors (EGCN):** This field indicates the number of Endurance Group Configuration Descriptors in the list. If this field is cleared to 0h, then no Endurance Group Configuration Descriptors are reported.
31:6	Reserved
NOTE 1	**Endurance Group Configuration 0 Descriptor:** This field indicates the first Endurance Group Configuration Descriptor (refer to Figure 252) in the list, if any.
NOTE 1	**Endurance Group Configuration 1 Descriptor:** This field indicates the second Endurance Group Configuration Descriptor in the list, if any.
…	
NOTE 1	**Endurance Group Configuration EGCN-1 Descriptor:** This field indicates the last Endurance Group Configuration Descriptor in the list, if any.
Notes: 1. Endurance Group Configuration Descriptors may be different lengths.	

The Endurance Group Configuration Descriptor is defined in Figure 252.

In the Endurance Group Configuration Descriptor (refer to Figure 252), NVM Set Identifiers shall be listed in ascending order by value, and each NVM Set Identifier shall appear only once.

In the Endurance Group Configuration Descriptor, Channel Configuration Descriptors shall be listed in ascending order by Channel Identifier value, and each Channel Identifier shall appear only once.

Figure 252: Endurance Group Configuration Descriptor

Bytes	Description
1:0	**Endurance Group Identifier (ENDGID):** This field indicates the identifier of the Endurance Group (refer to section 3.2.3) described by this Endurance Group Configuration Descriptor. This field shall indicate a value greater than or equal to 1h and less than or equal to the value of the Endurance Group Identifier Maximum field (refer to Figure 275).
3:2	**Capacity Adjustment Factor:** This field indicates the capacity adjustment factor (refer to section 8.3.1) for this Endurance Group. A value of FFFFh indicates that value and all higher values. A value of 0h indicates that the Capacity Adjustment Factor is not reported.
15:4	Reserved
31:16	**Total Endurance Group Capacity (TEGCAP):** This field indicates the total NVM capacity in this Endurance Group. The value is in bytes. If this field is cleared to 0h, the NVM subsystem does not report the total NVM capacity in this Endurance Group.
47:32	**Spare Endurance Group Capacity (SEGCAP):** This field indicates the spare NVM capacity in this Endurance Group. The value is in bytes. If this field is cleared to 0h, the NVM subsystem does not report the unallocated NVM capacity in this Endurance Group.
63:48	**Endurance Estimate:** This field is an estimate of the total number of data bytes that may be written to the Endurance Group over the lifetime of the Endurance Group assuming a write amplification of 1 (i.e., no increase in the number of write operations performed by the device beyond the number of write operations requested by a host). This value is reported in billions (i.e., a value of 1h corresponds to 1,000,000,000 bytes written) and is rounded up (e.g., a value of 1h indicates the number of bytes written is from 1 to 1,000,000,000, 2h indicates the number of bytes written is from 1,000,000,001 to 2,000,000,000). A value of FFFFFFFF_FFFFFFFF_FFFFFFFF_FFFFFFFFh means that value and all higher values. A value of 0h indicates that the Endurance Estimate is not reported. The relationship between this value and the value in the Endurance Estimate field in the Endurance Group Information log page (refer to section 5.14.1.9) is outside the scope of this specification.
79:64	Reserved
NVM Set Identifiers	
81:80	**Number of NVM Sets (EGSETS):** This field indicates the number of NVM Set Identifiers in this Endurance Group Configuration Descriptor. A value of 0h indicates that no NVM Set Identifiers are reported for this Endurance Group.
83:82	**NVM Set 0 Identifier:** This field indicates the identifier of the first NVM Set assigned to this Endurance Group, if reported. Refer to section 3.2.2.
85:84	**NVM Set 1 Identifier:** This field indicates the identifier of the second NVM Set assigned to this Endurance Group, if reported.
	…
(EGSETS*2)+81 : (EGSETS*2)+80	**NVM Set EGSETS-1 Identifier:** This field indicates the identifier of the last NVM Set assigned to this Endurance Group, if reported.
Channel Configuration Descriptors	
(EGSETS*2)+83 : (EGSETS*2)+82	**Number of Channels (EGCHANS):** This field indicates the number of Channel Configuration Descriptors in this Endurance Group Configuration Descriptor. If this field is cleared to 0h, then no Channel Configuration Descriptors are reported for this Endurance Group.
NOTE 1	**Channel 0 Configuration Descriptor:** This field contains the Channel Configuration Descriptor (refer to Figure 253) for the first Channel in this Endurance Group, if any.
NOTE 1	**Channel 1 Configuration Descriptor:** This field contains the Channel Configuration Descriptor for the second Channel in this Endurance Group, if any.
	…

Figure 252: Endurance Group Configuration Descriptor

Bytes	Description
NOTE 1	**Channel EGCHANS-1 Configuration Descriptor:** This field contains the Channel Configuration Descriptor for the last Channel in this Endurance Group, if any.
Notes: 1. Channel Configuration Descriptors may be different lengths.	

The Channel Configuration Descriptor (refer to Figure 253) lists the Media Units attached to a Channel. Media Unit Configuration Descriptors (refer to Figure 254) shall be listed in ascending order by Media Unit Identifier, and each Media Unit Identifier shall appear only once.

Figure 253: Channel Configuration Descriptor

Byte	Description
1:0	**Channel Identifier:** This field indicates the identifier of this Channel. A value of FFFFh indicates that the Channel Identifier is not specified.
3:2	**Number of Channel Media Units (CHMUS):** This field indicates the number of Media Units that are attached to this Channel. If this field is cleared to 0h, then no Media Unit Configuration Descriptors are reported for this Channel.
NOTE 1	**Media Unit 0 Configuration Descriptor:** This field contains the Media Unit Configuration Descriptor (refer to Figure 254) for the first Media Unit attached to this Channel, if any.
NOTE 1	**Media Unit 1 Configuration Descriptor:** This field contains the Media Unit Configuration Descriptor for the second Media Unit attached to this Channel, if reported.
	…
NOTE 1	**Media Unit CHMUS-1 Configuration Descriptor:** This field contains the Media Unit Configuration Descriptor for the last Media Unit attached to this Channel, if any.
Notes: 1. Media Unit Configuration Descriptors may be different lengths.	

The Media Unit Configuration Descriptor is defined in Figure 254.

Figure 254: Media Unit Configuration Descriptor

Byte	Description
1:0	**Media Unit Identifier:** This field indicates the identifier of this Media Unit.
5:2	Reserved
7:6	**Media Unit Descriptor Length (MUDL):** This field contains the length in bytes of the descriptor information that follows. The total length of the Media Unit Configuration Descriptor in bytes is the value in this field plus 8. This field shall be cleared to 0h.

5.16.1.18 Feature Identifiers Supported and Effects (Log Identifier 12h)

An NVM subsystem may support several interfaces for submitting a Get Log Page command such as an Admin Submission Queue, PCIe VDM Management Endpoint, or SMBus/I2C Management Endpoint (refer the NVM Express Management Interface Specification for details on Management Endpoints) and may have zero or more instances of each of those interfaces. The feature identifiers (FIDs) supported on each instance of each interface may be different. This log page describes the FIDs that are supported on the interface to which the Get Log Page command was submitted and the effects of those features on the state of the NVM subsystem. The log page is defined in Figure 255. Each Feature Identifier's effects are described in a FID Supported and Effects data structure defined in Figure 256.

If the UUID Selection Supported bit is set to '1' for the Get Log Page command in the Commands Supported and Effects log page (refer to section 5.16.1.6), then the log page data reflects the FIDs that are supported based on the value of the UUID Index field (refer to section 8.25).

The features that the controller supports are dependent on the I/O Command Set that is based on:

- the I/O Command Set selected in CC.CSS, if CC.CSS is not set to 110b; and
- the Command Set Identifier (CSI) field in CDW 14, if CC.CSS is set to 110b.

If CC.CSS is set to 110b, I/O Command Sets that have not been enabled by the I/O Command Set Profile (FID 19h) (refer to section 5.27.1.21) are treated as unsupported.

Figure 255: Feature Identifiers Effects Log Page

Bytes	Description
03:00	**Feature Identifier Supported 0:** Contains the FID Supported and Effects data structure (refer to Figure 256) for FID 0h.
07:04	**Feature Identifier Supported 1:** Contains the FID Supported and Effects data structure (refer to Figure 256) for FID 1h.
…	…
1019:1016	**Feature Identifier Supported 254:** Contains the FID Supported and Effects data structure (refer to Figure 256) for FID FEh.
1023:1020	**Feature Identifier Supported 255:** Contains the FID Supported and Effects data structure (refer to Figure 256) for FID FFh.

The FID Supported and Effects data structure describes the effect of a Set Features command for the FID, including any optional features of the FID.

Figure 256: FID Supported and Effects Data Structure

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:20</td><td>FID Scope (FSP): This field defines the scope for the associated feature identifier. If the value of this field is 0h, then no scope is reported. If this field is non-zero, then only one bit shall be set to ‘1’.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>11:6</td><td>Reserved</td></tr>
<tr><td>5</td><td>NVM Subsystem Scope: If set to ‘1’, then modifying the value of the FID may impact the whole NVM subsystem. If cleared to ‘0’ and the FSP field is non-zero, then modifying the value of the FID does not impact the whole NVM subsystem.</td></tr>
<tr><td>4</td><td>Domain Scope: If set to ‘1’, then modifying the value of the FID may impact a single Domain. If cleared to ‘0’ and the FSP field is non-zero, then modifying the value of the FID does not impact a single Domain.</td></tr>
<tr><td>3</td><td>Endurance Group Scope: If set to ‘1’, then modifying the value of the FID may impact Endurance Groups. If cleared to ‘0’ and the FSP field is non-zero, then modifying the value of the FID does not impact Endurance Groups.</td></tr>
<tr><td>2</td><td>NVM Set Scope: If set to ‘1’, then modifying the value of the FID may impact NVM Sets. If cleared to ‘0’ and the FSP field is non-zero, then modifying the value of the FID does not impact NVM Sets.</td></tr>
<tr><td>1</td><td>Controller Scope: If set to ‘1’, then modifying the value of the FID may impact the controller. If cleared to ‘0’ and the FSP field is non-zero, then the FID does not have controller scope.</td></tr>
<tr><td>0</td><td>Namespace Scope: If set to ‘1’, then modifying the value of the FID may impact namespaces. If cleared to ‘0’ and the FSP field is non-zero, then modifying the value of the FID does not impact namespaces.</td></tr>
</table></td></tr>
<tr><td>19</td><td>UUID Selection Supported: If set to ‘1’, then the controller supports the selection of a UUID (refer to section 8.25) by a Get Features command or a Set Features command using this FID. If cleared to ‘0’, then the controller does not support the selection of a UUID by a Get Features command or a Set Features command using this FID.</td></tr>
<tr><td>18:05</td><td>Reserved</td></tr>
<tr><td>04</td><td>Controller Capability Change (CCC): If this bit is set to ‘1’, then changing the value of this FID may change controller capabilities. If this bit is cleared to ‘0’, then changing the value of this FID does not modify controller capabilities. Controller capability changes include a firmware update that changes the capabilities reported in the CAP property.</td></tr>
</table>

Figure 256: FID Supported and Effects Data Structure

Bits	Description
03	**Namespace Inventory Change (NIC):** If this bit is set to '1', then changing the value of this FID may change the number of namespaces or capabilities for multiple namespaces. If this bit is cleared to '0', then changing the value of this FID does not modify the number of namespaces or capabilities for multiple namespaces. Namespace inventory changes include adding or removing namespaces.
02	**Namespace Capability Change (NCC):** If this bit is set to '1', then changing the value of this FID may change the capabilities of a single namespace. If this bit is cleared to '0', then changing the value of this FID does not modify any namespace capabilities for the specified namespace. Namespace capability changes include a logical format change.
01	**User Data Content Change (UDCC):** If this bit is set to '1', then changing the value of this FID may modify user data content in one or more namespaces. If this bit is cleared to '0', then changing the value of this FID does not modify user data content in any namespace.
00	**FID Supported (FSUPP):** If this bit is set to '1', then this FID is supported by the controller. If this bit is cleared to '0', then this FID is not supported by the controller and all other fields in this structure shall be cleared to 0h. Refer to section 3.1.2 for the FID support requirements for each controller type.

5.16.1.19 NVMe-MI Commands Supported and Effects (Log Identifier 13h)

This log page describes the Management Interface Command Set commands (refer to the NVM Express Management Interface Specification) that the controller supports using the NVMe-MI Send and NVMe-MI Receive commands and the effects of those Management Interface Command Set commands on the state of the NVM subsystem. The log page is defined in Figure 257.

Figure 257: NVMe-MI Commands Supported and Effects Log Page

Bytes	Description
03:00	**Management Interface Command Supported 0:** Contains the NVMe-MI Commands Supported and Effects data structure (refer to Figure 258) for the Management Interface command with an opcode value of 0h.
07:04	**Management Interface Command Supported 1:** Contains the NVMe-MI Commands Supported and Effects data structure (refer to Figure 258) for the Management Interface command with an opcode value of 1h.
…	…
1019:1016	**Management Interface Command Supported 254:** Contains the NVMe-MI Commands Supported and Effects data structure (refer to Figure 258) for the Management Interface command with an opcode value of 254.
1023:1020	**Management Interface Command Supported 255:** Contains the NVMe-MI Commands Supported and Effects data structure (refer to Figure 258) for the Management Interface command with an opcode value of 255.
4095:1024	Reserved

The NMVe-MI Commands Supported and Effects data structure describes the overall possible effect of a Management Interface command using the using the NVMe-MI Send command, including any optional features of the command.

Figure 258: NVMe-MI Commands Supported and Effects Data Structure

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:20</td><td>Command Scope (CSP): This field defines the scope for the associated NVMe-MI Send command that specifies the Management Interface command opcode for this data structure. If the value of this field is 0h, then no scope is reported. If this field is non-zero, then only one bit shall be set to ‘1’.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>11:6</td><td>Reserved</td></tr>
<tr><td>5</td><td>NVM Subsystem Scope: If set to ‘1’, then the command performs actions that may impact the whole NVM subsystem. If cleared to ‘0’ and the CSP field is non-zero, then the command performs actions that do not impact the whole NVM subsystem.</td></tr>
<tr><td>4</td><td>Domain Scope: If set to ‘1’, then the command has Domain scope. If cleared to ‘0’ and the CSP field is non-zero, then the command does not have Domain scope.</td></tr>
<tr><td>3</td><td>Endurance Group Scope: If set to ‘1’, then the command has Endurance Group scope. If cleared to ‘0’ and the CSP field is non-zero, then the command does not have Endurance Group scope.</td></tr>
<tr><td>2</td><td>NVM Set Scope: If set to ‘1’, then the command has NVM Set scope. If cleared to ‘0’ and the CSP field is non-zero, then the command does not have NVM Set scope.</td></tr>
<tr><td>1</td><td>Controller Scope: If set to ‘1’, then the command has controller scope. If cleared to ‘0’ and the CSP field is non-zero, then the command does not have controller scope.</td></tr>
<tr><td>0</td><td>Namespace Scope: If set to ‘1’, then the command has namespace scope. If cleared to ‘0’ and the CSP field is non-zero, then the command does not have namespace scope.</td></tr>
</table></td></tr>
<tr><td>19:05</td><td>Reserved</td></tr>
<tr><td>04</td><td>Controller Capability Change (CCC): If this bit is set to ‘1’, then this command may change controller capabilities. If this bit is cleared to ‘0’, then this command does not modify controller capabilities. Controller capability changes include a firmware update that changes the capabilities reported in the CAP property.</td></tr>
<tr><td>03</td><td>Namespace Inventory Change (NIC): If this bit is set to ‘1’, then this command may change the number of namespaces or capabilities for multiple namespaces. If this bit is cleared to ‘0’, then this command does not modify the number of namespaces or capabilities for multiple namespaces. Namespace inventory changes include adding or removing namespaces.</td></tr>
<tr><td>02</td><td>Namespace Capability Change (NCC): If this bit is set to ‘1’, then this command may change the capabilities of a single namespace. If this bit is cleared to ‘0’, then this command does not modify any namespace capabilities for the specified namespace. Namespace capability changes include a logical format change.</td></tr>
<tr><td>01</td><td>User Data Content Change (UDCC): If this bit is set to ‘1’, then this command may modify user data content in one or more namespaces. If this bit is cleared to ‘0’, then this command does not modify user data content in any namespace. User data content changes include a write operation.</td></tr>
<tr><td>00</td><td>Command Supported (CSUPP): If this bit is set to ‘1’, then this command is supported by the controller. If this bit is cleared to ‘0’, then this command is not supported by the controller and all other fields in this structure shall be cleared to 0h.</td></tr>
</table>

5.16.1.20 Command and Feature Lockdown (Log Identifier 14h)

This log page is used to indicate which commands and Set Features Feature Identifiers are supported to be prohibited from execution using the Command and Feature Lockdown capability (refer to section 8.4) and which commands are currently prohibited if received on an NVM Express controller Admin Submission Queue or received out-of-band on a Management Endpoint (refer to the NVM Express Management Interface Specification). This log page uses the Log Specific Parameter field in Command Dword 10 (refer to Figure 197) as defined in Figure 259. This log page may use the UUID Index field in the Get Log Page command to specify the scope and content of the list returned in the Command and Feature Identifier List field of this log page. The UUID Index field may be used if the Scope field is set to 2h, allowing returning of vendor specific Set Features Feature Identifier lockdown information.

Figure 259: Command and Feature Lockdown Log Specific Parameter Field

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>14</td><td>Reserved</td></tr>
<tr><td>13:12</td><td>Contents (CNTTS): This field in combination with the Scope field specifies the contents of the Command and Feature Identifier List field in the log page.
<table>
<tr><th>Value</th><th>Command and Feature Identifier List Definition</th></tr>
<tr><td>00b</td><td>List of command opcodes or Feature Identifiers based on the Scope field that are supported to be prohibited.</td></tr>
<tr><td>01b</td><td>List of command opcodes or Feature Identifiers based on the Scope field that are currently prohibited if received on an NVM Express controller Admin submission queue.</td></tr>
<tr><td>10b</td><td>List of command opcodes or Feature Identifiers based on the Scope field that are currently prohibited if received out-of-band on a Management Endpoint.</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>11:08</td><td>Scope (SCP): This field in combination with the Contents field specifies the contents of the Command and Feature Identifier List field in the log page.
<table>
<tr><th>Value</th><th>Command and Feature Identifier List Contents</th></tr>
<tr><td>0h</td><td>List of Admin Command Set opcodes</td></tr>
<tr><td>1h</td><td>Reserved</td></tr>
<tr><td>2h</td><td>List of Feature Identifiers</td></tr>
<tr><td>3h</td><td>List of a Management Interface Command Set opcodes (refer to the NVM Express Management Interface Specification)</td></tr>
<tr><td>4h</td><td>List of a PCIe Command Set opcodes (refer to the NVM Express Management Interface Specification)</td></tr>
<tr><td>5h to Fh</td><td>Reserved</td></tr>
</table></td></tr>
</table>

If a UUID Index is specified in the Get Log Page command (refer to section 5.16) with the Scope field is set to 2h, then the controller should return vendor specific Set Features lockdown information defined by the vendor identified by the specified UUID index field. If the Scope field is not set to 2h, then the UUID index field is ignored.

If a controller processes this command with the Contents field set to 10b and the NVM subsystem does not contain a Management Endpoint, then the command shall be aborted with a status code of Invalid Field in Command.

The log page returned is defined in Figure 260.

Figure 260: Command and Feature Lockdown Log Page

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:6</td><td>Reserved</td></tr>
<tr><td>5:4</td><td>Contents Selected (CS): This field in combination with the Scope Selected field indicates the contents of the Command and Feature Identifier List field in the log page. The Content Selected field is specified by the contents of the Contents field in the Log Specific Parameter field of the Get Log Page command.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>00b</td><td>List contains command opcodes or Set Features Feature Identifiers based on the Scope Selected field that are supported to be prohibited</td></tr>
<tr><td>01b</td><td>List contains command opcodes or Set Features Feature Identifiers based on the Scope Selected field that are currently prohibited if received on an NVM Express controller submission queue</td></tr>
<tr><td>10b</td><td>List contains command opcodes or Set Features Feature Identifiers based on the Scope field that are currently prohibited if received out-of-band on a Management Endpoint</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>3:0</td><td>Scope Selected (SS): This field in combination with the Contents Selected field indicates what the Command and Feature Identifier List field contains in the log page. The Scope Selected field is specified by the contents of the Scope field in the Log Specific Parameter field of the Get Log Page command.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>List contains Admin Command Set opcodes</td></tr>
<tr><td>1h</td><td>Reserved</td></tr>
<tr><td>2h</td><td>List contains Set Features Feature Identifiers</td></tr>
<tr><td>3h</td><td>List contains Management Interface Command Set opcodes</td></tr>
<tr><td>4h</td><td>List contains PCIe Command Set opcodes</td></tr>
<tr><td>5h to Fh</td><td>Reserved</td></tr>
</table></td></tr>
</table>
</td></tr>
<tr><td>2:1</td><td>Reserved</td></tr>
<tr><td>3</td><td>Length (LNGTH): This field indicates the length in bytes (n) of the Command and Feature Identifier List field that follow in the log page. If the Command and Feature Identifier List field contains no coded values, then this field shall be cleared to 0h.</td></tr>
<tr><td>n+3:4</td><td>Command and Feature Identifier List (CFIL): The contents of this field are dependent on the setting of the Contents Selected field and Scope Selected field. This field contains a list of coded values identified by the Scope Selected field and the Content Selected field. The list shall be in order from lowest numerical value to highest numerical value.</td></tr>
<tr><td>511:n+4</td><td>Reserved</td></tr>
</table>

5.16.1.21 Boot Partition (Log Identifier 15h)

The Boot Partition log page provides read only access to the Boot Partition (refer to section 8.2) accessible by this controller through the BPRSEL register (refer to section 3.1.3.14).

This log consists of a header describing the Boot Partition and Boot Partition data as defined by Figure 262. The Boot Partition Identifier bit in the Log Specific Parameter field determines the Boot Partition.

A host reading this log page has no effects on the BPINFO (refer to section 3.1.3.13), BPRSEL, and BPMBL (refer to section 3.1.3.15) registers.

The Log Specific Parameter field in Command Dword 10 (refer to Figure 197) for this log page is defined in Figure 261.

Figure 261: Boot Partition Log Specific Parameter Field

Bits	Description
14:09	Reserved

Figure 261: Boot Partition Log Specific Parameter Field

Bits	Description
08	**Boot Partition Identifier:** This bit specifies the Boot Partition identifier for the Boot Partition to return.

Figure 262: Boot Partition Log Page

Bytes	Description
Boot Partition Header	
00	**Log Identifier:** This field shall be set to 15h.
03:01	Reserved
07:04	**Boot Partition Information:** Contains defines the characteristics of Boot Partitions. Bits: 31 — **Active Boot Partition ID (ABPID):** This bit indicates the identifier of the active Boot Partition. Bits: 30:15 — Reserved Bits: 14:00 — **Boot Partition Size (BPSZ):** This field defines the size of the Boot Partition Data field in multiples of 128 KiB.
15:08	Reserved
Boot Partition Data	
BPSZ*128 KiB + 15:16	**Boot Partition Data:** Contains the contents of the specified Boot Partition.

5.16.1.22 Rotational Media Information Log (Log Identifier 16h)

This log page provides rotational media information (refer to section 8.20) for Endurance Groups that store data on rotational media. The information provided is retained across power cycles and resets.

The Endurance Group Identifier is specified in the Log Specific Identifier field in Command Dword 11 of the Get Log Page command.

If the NVM subsystem does not contain any Endurance Groups that store data on rotational media, then the Rotational Media Information Log should not be supported.

Figure 263: Rotational Media Information Log Page

Bytes	Description
1:0	**Endurance Group Identifier:** The Endurance Group Identifier specified by the Get Log Page command.
3:2	**Number of Actuators:** Contains the number of actuators in this Endurance Group.
5:4	**Nominal Rotational Speed (NRS)**: Value: 0000h — Not reported Value: 0001h — This value is prohibited to maintain backward compatibility with other standards. This value shall not be used. Value: FFFFh — Reserved Value: All other values — Nominal rotational speed in revolutions per minute while the current Power State is 0 (refer to section 5.27.1.2).
7:6	Reserved
11:8	**Spinup Count**: Contains the total number of successful spinup events for this Endurance Group over the lifetime of the Endurance Group. If the Spinup Count is less than FFFFFFFFh, then the controller shall increment this count by one for each successful spinup event. A successful spinup event occurs when the controller power state transitions from a non-operational power state to an operational power state.

Figure 263: Rotational Media Information Log Page

Bytes	Description
15:12	**Failed Spinup Count**: Contains the total number of failed spinup events for this Endurance Group over the lifetime of the Endurance Group. If the Failed Spinup Count is less than FFFFFFFFh, then the controller shall increment this count by one for each failed spinup event. A failed spinup event occurs when the controller fails an attempt to transition from a non-operational power state to an operational power state.
19:16	**Load Count**: Contains the total number successful actuator load events for this Endurance Group over the lifetime of the Endurance Group. If the Load Count is less than FFFFFFFFh, then the controller shall increment this count by one for each successful actuator load event. A successful actuator load event occurs if an actuator transitions from a non-operational state to an operational state.
23:20	**Failed Load Count**: Contains the number of failed actuator load events for this Endurance Group over the lifetime of the Endurance Group. If the Failed Load Count is less than FFFFFFFFh, then the controller shall increment this count by one for each failed actuator load event. A failed actuator load event occurs if an actuator fails an attempt to transition from a non-operational state to an operational state.
511:24	Reserved

5.16.1.23 Discovery Log Page (Log Identifier 70h)

The Discovery Log Page shall only be supported by Discovery controllers. The Discovery Log Page shall not be supported by controllers that expose namespaces for NVMe over PCIe or NVMe over Fabrics. The Discovery Log Page provides an inventory of NVM subsystems with which a host may attempt to form an association. The Discovery Log Page may be specific to the host requesting the log. The Discovery Log Page is persistent across power cycles.

The Log Page Offset may be used to retrieve specific records. The number of records is returned in the header of the log page. The format for a Discovery Log Page Entry is defined in Figure 264. The format for the Discovery Log Page is defined in Figure 265.

A single Get Log Page command used to read the Discovery Log Page shall be atomic. If the host reads the Discovery Log Page using multiple Get Log Page commands the host should ensure that there has not been a change in the contents of the data. The host should read the Discovery Log Page contents in order (i.e., with increasing Log Page Offset values) and then re-read the Generation Counter after the entire log page is transferred. If the Generation Counter does not match the original value read, the host should discard the log page read as the entries may be inconsistent. If the log page contents change during this command sequence, the controller may return a status code of Discover Restart.

Every record indicates via the SUBTYPE field if that record is referring to another Discovery Service or if the record indicates an NVM subsystem composed of controllers that may expose namespaces. A referral to another Discovery Service (i.e., SUBTYPE 01h) is a mechanism to find additional Discovery subsystems. An NVM subsystem entry (i.e., SUBTYPE 02h) is a mechanism to find NVM subsystems that contain controllers that may expose namespaces. Referrals shall not be deeper than eight levels.

If an NVM subsystem supports the dynamic controller model, then all entries for that NVM subsystem shall have the Controller ID field set to FFFFh. For a particular NVM subsystem port and NVMe Transport address in an NVM subsystem, there shall be no more than one entry with the Controller ID field set to:

- FFFFh if that NVM subsystem supports the dynamic controller model; or
- FFFEh if that NVM subsystem supports the static controller model.

Figure 264: Discovery Log Page Entry Data Structure

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>00</td><td>Transport Type (TRTYPE): Specifies the NVMe Transport type.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00</td><td>Reserved</td></tr>
<tr><td>01</td><td>RDMA Transport (refer to the NVMe RDMA Transport specification)</td></tr>
<tr><td>02</td><td>Fibre Channel Transport (refer to INCITS 556)</td></tr>
<tr><td>03</td><td>TCP Transport (refer to the NVMe TCP Transport specification)</td></tr>
<tr><td>04 to 253</td><td>Reserved</td></tr>
<tr><td>254</td><td>Intra-host Transport (i.e., loopback) (NOTE: This is a reserved value for use by host software.)</td></tr>
<tr><td>255</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>01</td><td>Address Family (ADRFAM): Specifies the address family.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00</td><td>Reserved</td></tr>
<tr><td>01</td><td>AF_INET: IPv4 address family. IPv4address format syntax specified in section 3.2.2 of IETF RFC 3986.</td></tr>
<tr><td>02</td><td>AF_INET6: IPv6 address family. IPv6address format syntax specified in section 3.2.2 of IETF RFC 3986.</td></tr>
<tr><td>03</td><td>AF_IB: InfiniBand address family.</td></tr>
<tr><td>04</td><td>Fibre Channel address family.</td></tr>
<tr><td>05 to 253</td><td>Reserved</td></tr>
<tr><td>254</td><td>Intra-host Transport (i.e., loopback) (NOTE: This is a reserved value for use by host software.)</td></tr>
<tr><td>255</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>02</td><td>Subsystem Type (SUBTYPE): Specifies the type of the NVM subsystem that is indicated in this entry.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00</td><td>Reserved.</td></tr>
<tr><td>01</td><td>The entry describes a referral to another Discovery Service composed of Discovery controllers for additional records.</td></tr>
<tr><td>02</td><td>The entry describes an NVM subsystem that is not associated with Discovery controllers and whose controllers may have attached namespaces.</td></tr>
<tr><td>03 to 255</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>03</td><td>Transport Requirements (TREQ): Indicates requirements for the NVMe Transport.

Bits 7:3 are reserved.

Bit 2 if set to ‘1’ indicates that the controller is capable of disabling SQ flow control. A controller that is capable of disabling SQ flow control may accept or reject a host request to disable SQ flow control. If cleared to ‘0’, then the controller requires use of SQ flow control.

Bits 1:0 indicate whether connections shall be made over a fabric secure channel (which includes authentication) (refer to section 8.13).
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>Not specified</td></tr>
<tr><td>01b</td><td>Required</td></tr>
<tr><td>10b</td><td>Not required</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>05:04</td><td>Port ID (PORTID): Specifies a particular NVM subsystem port. Different NVMe Transports or address families may utilize the same Port ID value (e.g., a Port ID may support both iWARP and RoCE).</td></tr>
</table>

Figure 264: Discovery Log Page Entry Data Structure

Bytes	Description
07:06	**Controller ID (CNTLID):** Specifies the controller ID. If the NVM subsystem uses a dynamic controller model, then this field shall be set to FFFFh. If the NVM subsystem uses a static controller model, then this field may be set to a specific controller ID (values 0h to FFEFh are valid). If the NVM subsystem uses a static controller model and the value indicated is FFFEh, then the host should remember the Controller ID returned as part of the Fabrics Connect command in order to re-establish an association in the future with the same controller.
09:08	**Admin Max SQ Size (ASQSZ):** Specifies the maximum size of an Admin Submission Queue. This applies to all controllers in the NVM subsystem. The value shall be a minimum of 32 entries.
31:10	Reserved
63:32	**Transport Service Identifier (TRSVCID):** Specifies the NVMe Transport service identifier as an ASCII string. The NVMe Transport service identifier is specified by the associated NVMe Transport binding specification.
255:64	Reserved
511:256	**NVM Subsystem Qualified Name (SUBNQN):** NVMe Qualified Name (NQN) that uniquely identifies the NVM subsystem. Refer to section 4.4. For a Discovery Service, the value returned shall be the well-known Discovery Service NQN (nqn.2014-08.org.nvmexpress.discovery).
767:512	**Transport Address (TRADDR):** Specifies the address of the NVM subsystem that may be used for a Connect command as an ASCII string. The Address Family field describes the reference for parsing this field. Refer to section 1.4.2 for ASCII string requirements. For the definition of this field, refer to the appropriate NVMe Transport binding specification.
1023:768	**Transport Specific Address Subtype (TSAS):** Specifies NVMe Transport specific information about the address. For the definition of this field, refer to the appropriate NVMe Transport binding specification.

Figure 265: Discovery Log Page

Bytes	Description
07:00	**Generation Counter (GENCTR):** Indicates the version of the discovery information, starting at a value of 0h. For each change in the Discovery Log Page, this counter is incremented by one. If the value of this field is FFFFFFFF_FFFFFFFFh, then the field shall be cleared to 0h when incremented (i.e., rolls over to 0h).
15:08	**Number of Records (NUMREC):** Indicates the number of records contained in the log.
17:16	**Record Format (RECFMT):** Specifies the format of the Discovery Log Page. If a new format is defined, this value is incremented by one. The format of the record specified in this definition shall be 0h.
1023:18	Reserved
2047:1024	**Discovery Log Page Entry 0 (DLE0):** Contains the first Discovery Log Page Entry as defined in Figure 264.
3071:2048	**Discovery Log Page Entry 1 (DLE1):** Contains the second Discovery Log Page Entry as defined in Figure 264 (if present).
…	…
(((N + 2) × 1024) - 1): ((N + 1) × 1024)	**Discovery Log Page Entry N (DLEN):** Contains the Nth Discovery Log Page Entry as defined in Figure 264 (if present).

5.16.1.24 Reservation Notification (Log Identifier 80h)

The Reservation Notification log page reports one log page from a time ordered queue of Reservation Notification log pages, if available. A new Reservation Notification log page is created and added to the end of the queue of reservation notifications whenever an unmasked reservation notification occurs on any namespace that is attached to the controller. The Get Log Page command:

- returns a data buffer containing a log page corresponding to the oldest log page in the reservation notification queue (i.e., the log page containing the lowest Log Page Count field; accounting for wrapping); and
- removes that Reservation Notification log page from the queue.

If there are no available Reservation Notification log page entries when a Get Log Page command is issued, then an empty log page (i.e., all fields in the log page cleared to 0h) shall be returned.

If the controller is unable to store a reservation notification in the Reservation Notification log page due to the size of the queue, that reservation notification is lost. If a reservation notification is lost, then the controller shall increment the Log Page Count field of the last reservation notification in the queue (i.e., the Log Page Count field in the last reservation notification in the queue shall contain the value associated with the most recent reservation notification that has been lost).

The format of the log page is defined in Figure 266.

Figure 266: Reservation Notification Log Page

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>07:00</td><td>Log Page Count: This is a 64-bit incrementing Reservation Notification log page count, indicating a unique identifier (modulo 64 bit) for this notification. The count starts at 0h following a Controller Level Reset and is incremented for every event that causes a reservation notification regardless of whether that notification is added to the queue. If the value of this field is FFFFFFFF_FFFFFFFFh, then the field is set to 1h when incremented (i.e., rolls over to 1h) and a new log page is created.

If there are no Reservation Notification log pages to return (i.e., the queue of Reservation Notification log pages is empty), then this field shall return the value 0h. Subsequent reservation notifications continue incrementing this unique identifier from the last non-zero value (i.e., the value that identified the previous Reservation Notification log page). A value of 0h indicates the log page is empty.</td></tr>
<tr><td>08</td><td>Reservation Notification Log Page Type: This field indicates the Reservation Notification type described by this log page.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0h</td><td>Empty Log Page: Get Log Page command was processed when no unread Reservation Notification log pages were available. All the fields of an empty log page shall have a value of 0h.</td></tr>
<tr><td>1h</td><td>Registration Preempted</td></tr>
<tr><td>2h</td><td>Reservation Released</td></tr>
<tr><td>3h</td><td>Reservation Preempted</td></tr>
<tr><td>4h to FFh</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>09</td><td>Number of Available Log Pages: This field indicates the number of additional available Reservation Notification log pages (i.e., the number of unread log pages not counting this one). If there are more than 255 additional available log pages, then a value of 255 is returned. A value of 0h indicates that there are no additional available log pages.</td></tr>
<tr><td>11:10</td><td>Reserved</td></tr>
<tr><td>15:12</td><td>Namespace ID: This field indicates the namespace ID of the namespace associated with the Reservation Notification described by this log page.</td></tr>
<tr><td>63:16</td><td>Reserved</td></tr>
</table>

5.16.1.25 Sanitize Status (Log Identifier 81h)

The Sanitize Status log page is used to report sanitize operation time estimates and information about the most recent sanitize operation (refer to section 8.20). The Get Log Page command returns a data buffer containing a log page formatted as defined in Figure 267. This log page shall be retained across power cycles and resets. This log page shall contain valid data whenever CSTS.RDY is set to ‘1’.

If the Sanitize Capabilities (SANICAP) field in the Identify Controller data structure is not cleared to 0h (i.e., the Sanitize command is supported), then this log page shall be supported. If the Sanitize Capabilities field in the Identify Controller data structure is cleared to 0h, then this log page is reserved.

Figure 267: Sanitize Status Log Page

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>01:00</td><td>Sanitize Progress (SPROG): This field indicates the fraction complete of the sanitize operation. The value is a numerator of the fraction complete that has 65,536 (10000h) as its denominator. This value shall be set to FFFFh if bits 2:0 of the SSTAT field are not set to 010b.

If a sanitize operation has been started by a Sanitize command with the No-Deallocate After Sanitize bit set to ‘1’ (refer to section 5.24) and if NODMMAS field in the Identify Controller data structure is set to 10b (refer to Figure 275), then the fraction reported shall include the time related to the additional media modification.</td></tr>
<tr><td>03:02</td><td>Sanitize Status (SSTAT): This field indicates the status associated with the most recent sanitize operation.

Bits 15:9 are reserved.

Bit 8 (Global Data Erased): If set to ‘1’, then no namespace user data in the NVM subsystem has been written to and no Persistent Memory Region in the NVM subsystem has been enabled:

a) since being manufactured and the NVM subsystem has never been sanitized; or
b) since the most recent successful sanitize operation.

If cleared to ‘0’, then a namespace user data in the NVM subsystem has been written to or a Persistent Memory Region in the NVM subsystem has been enabled:

a) since being manufactured and the NVM subsystem has never been sanitized; or
b) since the most recent successful sanitize operation of the NVM subsystem.

Bits 7:3 contains the number of completed passes if the most recent sanitize operation was an Overwrite. This field shall be cleared to 0h if the most recent sanitize operation was not an Overwrite.

Bits 2:0 contains the status of the most recent sanitize operation as shown below.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>The NVM subsystem has never been sanitized.</td></tr>
<tr><td>001b</td><td>The most recent sanitize operation completed successfully including any additional media modification (refer to the No-Deallocate Modifies Media After Sanitize field in Figure 275).</td></tr>
<tr><td>010b</td><td>A sanitize operation is currently in progress.</td></tr>
<tr><td>011b</td><td>The most recent sanitize operation failed.</td></tr>
<tr><td>100b</td><td>The most recent sanitize operation for which No-Deallocate After Sanitize (refer to section 5.24) was requested has completed successfully with deallocation of all user data (refer to section 5.27.1.19).</td></tr>
<tr><td>101b to 111b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>07:04</td><td>Sanitize Command Dword 10 Information (SCDW10): This field contains the value of the Command Dword 10 field of the Sanitize command that started the sanitize operation whose status is reported in the SSTAT field. Refer to Figure 303.</td></tr>
<tr><td>11:08</td><td>Estimated Time For Overwrite: This field indicates the number of seconds required to complete an Overwrite sanitize operation with 16 passes in the background (refer to section 5.24) when the No-Deallocate Modifies Media After Sanitize field (refer to Figure 275) is not set to 10b. A value of 0h indicates that the sanitize operation is expected to be completed in the background when the Sanitize command that started that operation is completed. A value of FFFFFFFFh indicates that no time period is reported.</td></tr>
<tr><td>15:12</td><td>Estimated Time For Block Erase: This field indicates the number of seconds required to complete a Block Erase sanitize operation in the background (refer to section 5.24) when the No-Deallocate Modifies Media After Sanitize field (refer to Figure 275) is not set to 10b. A value of 0h indicates that the sanitize operation is expected to be completed in the background when the Sanitize command that started that operation is completed. A value of FFFFFFFFh indicates that no time period is reported.</td></tr>
</table>

Figure 267: Sanitize Status Log Page

Bytes	Description
19:16	**Estimated Time For Crypto Erase:** This field indicates the number of seconds required to complete a Crypto Erase sanitize operation in the background (refer to section 5.24) when the No-Deallocate Modifies Media After Sanitize field (refer to Figure 275) is not set to 10b. A value of 0h indicates that the sanitize operation is expected to be completed in the background when the Sanitize command that started that operation is completed. A value of FFFFFFFFh indicates that no time period is reported.
23:20	**Estimated Time For Overwrite With No-Deallocate Media Modification:** This field indicates the number of seconds required to complete an Overwrite sanitize operation and the associated additional media modification after the Overwrite sanitize operation in the background (refer to section 5.24) when: a) the No-Deallocate After Sanitize bit was set to '1' in the Sanitize command that requested the Overwrite sanitize operation; and b) the No-Deallocate Modifies Media After Sanitize field (refer to Figure 275) is set to 10b. A value of 0h indicates that the sanitize operation is expected to be completed in the background when the Sanitize command that started that operation is completed. A value of FFFFFFFFh indicates that no time period is reported.
27:24	**Estimated Time For Block Erase With No-Deallocate Media Modification:** This field indicates the number of seconds required to complete a Block Erase sanitize operation and the associated additional media modification after the Block Erase sanitize operation in the background (refer to section 5.24) when: a) the No-Deallocate After Sanitize bit was set to '1' in the Sanitize command that requested the Block Erase sanitize operation; and b) the No-Deallocate Modifies Media After Sanitize field (refer to Figure 275) is set to 10b. A value of 0h indicates that the sanitize operation is expected to be completed in the background when the Sanitize command that started that operation is completed. A value of FFFFFFFFh indicates that no time period is reported.
31:28	**Estimated Time For Crypto Erase With No-Deallocate Media Modification:** This field indicates the number of seconds required to complete a Crypto Erase sanitize operation and the associated additional media modification after the Crypto Erase sanitize operation in the background (refer to section 5.24) when: a) the No-Deallocate After Sanitize bit was set to '1' in the Sanitize command that requested the Crypto Erase sanitize operation; and b) the No-Deallocate Modifies Media After Sanitize field (refer to Figure 275) is set to 10b. A value of 0h indicates that the sanitize operation is expected to be completed in the background when the Sanitize command that started that operation is completed. A value of FFFFFFFFh indicates that no time period is reported.
511:32	Reserved

5.16.2 Command Completion

Upon completion of the Get Log Page command, the controller posts a completion queue entry to the Admin Completion Queue. Get Log Page command specific status values are defined in Figure 268.

Figure 268: Get Log Page – Command Specific Status Values

Value	Description
9h	**Invalid Log Page:** The log page indicated is invalid or not supported. This error condition is also returned if a reserved log page is requested. Controllers compliant with NVM Express Base Specification revision 2.0 and earlier may return Invalid Field in Command for this condition.
29h	**I/O Command Set Not Supported:** The specified I/O Command Set is not supported by the controller.

5.17 Identify command

5.17.1 Identify command overview

The Identify command returns a data buffer that describes information about the NVM subsystem, the domain, the controller or the namespace(s). The data structure is 4,096 bytes in size.

The Identify command uses the Data Pointer, Command Dword 10, Command Dword 11, and Command Dword 14 fields. All other command specific fields are reserved.

Figure 269: Identify – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 87 for the definition of this field. If using PRPs, this field shall not be a pointer to a PRP List as the data buffer may not cross more than one page boundary.

Figure 270: Identify – Command Dword 10

Bits	Description
31:16	**Controller Identifier (CNTID):** This field specifies the controller identifier used as part of some Identify operations. Whether the CNTID field is used for a particular Identify operation is indicated in Figure 273. If this field is not used as part of the Identify operation, then: • host software shall clear this field to 0h for backwards compatibility (0h is a valid controller identifier); and • the controller shall ignore this field. Controllers that support the Namespace Management capability (refer to section 8.11) shall support this field.
15:08	Reserved
07:00	**Controller or Namespace Structure (CNS):** This field specifies the information to be returned to the host. Refer to Figure 273.

Figure 271: Identify – Command Dword 11

Bits	Description
31:24	**Command Set Identifier (CSI):** This field is CNS value specific. This field specifies the I/O Command Set to be used by the command for CNS values that require a Command Set Identifier. Refer to Figure 273 for Identify command CNS values that use this field. This field shall be cleared to 0h for Identify operations with CNS values that do not use this field. Values for this field are defined by Figure 274.
23:16	Reserved
15:00	**CNS Specific Identifier:** This field specifies an identifier that is required for a particular CNS value. The CNS values that require a CNS specific identifier are indicated in the table below.

CNS Value	Definition
NVM Set List (04h)	NVM Set Identifier (NVMSETID) (refer to section 3.2.2)
Domain List (18h)	Domain Identifier (DOMID) (refer to section 3.2.4.3)
Endurance Group List (19h)	Endurance Group Identifier (ENDGID) (refer to section 3.2.3)

If the controller supports selection of a UUID by the Identify command (refer to section 8.25), then Command Dword 14 is used to specify a UUID Index value (refer to Figure 272).

Figure 272: Identify – Command Dword 14

Bits	Description
31:07	Reserved
06:00	**UUID Index:** Refer to Figure 477.

The data structure returned is based on the Controller or Namespace Structure (CNS) field as shown in Figure 273. If there are fewer entries to return for the data structure indicated based on CNS value, then the unused portion of the returned data is zero filled. If a controller does not support the specified CNS value, then the controller shall abort the command with a status code of Invalid Field in Command.

When issuing the Identify command, if the specified namespace is not associated with an I/O Command Set that supports the specified Identify CNS value (refer to Figure 273), then the controller shall abort the command with a status code of Invalid I/O Command Set.

Note: The CNS field was specified as a one bit field in revision 1.0 and is a two bit field in revision 1.1. Host software should only issue CNS values defined in revision 1.0 to controllers compliant with revision 1.0. Host software should only issue CNS values defined in revision 1.1 to controllers compliant with revision 1.1. The results of issuing other CNS values to controllers compliant with revision 1.0 or revision 1.1, respectively, are indeterminate.

The Identify Controller data structure, Identify Namespace data structure, and the I/O Command Set specific Identify Namespace data structure include several unique identifiers. The format and layout of these unique identifiers is described in section 4.5.1.

Figure 273: Identify – CNS Values

CNS Value	O/M[1]	Definition	NSID[2]	CNTID[3]	CSI[4]	Reference Section
Active Namespace Management						
00h	M[11]	Identify Namespace data structure for the specified NSID or the common namespace capabilities for the NVM Command Set.[7]	Y	N	N[8]	NVM Command Set Specification
01h	M	Identify Controller data structure for the controller processing the command.[7]	N	N	N	5.17.2.1
02h	M	Active Namespace ID list.	Y	N	N	5.17.2.2
03h	M	Namespace Identification Descriptor list for the specified NSID.	Y	N	N	5.17.2.3
04h	O	An NVM Set List (refer to Figure 278) is returned to the host for up to 31 NVM Sets. The list contains entries for NVM Set identifiers greater than or equal to the value specified in the NVM Set Identifier (CDW11.NVMSETID) field.	N	N	N	5.17.2.4
05h	M	I/O Command Set specific Identify Namespace data structure for the specified NSID for the I/O Command Set specified in the CSI field.[7]	Y	N	Y	5.17.2.5
06h	M	I/O Command Set specific Identify Controller data structure for the controller processing the command.[7]	N	N	Y	5.17.2.6
07h	M	Active Namespace ID list associated with the specified I/O Command Set.	Y	N	Y	5.17.2.7
08h	M	I/O Command Set Independent Identify Namespace data structure.	Y	N	N	5.17.2.8
09h to 0Fh		Reserved				
Controller and Namespace Management						
10h	O[5]	Allocated Namespace ID list.	Y	N	N	5.17.2.9
11h	O[5, 11]	Identify Namespace data structure for the specified allocated NSID.	Y	N	N[8]	5.17.2.10
12h	O[5]	Controller List of controllers attached to the specified NSID.	Y	Y	N	5.17.2.11
13h	O[5]	Controller List of controllers that exist in the NVM subsystem.	N	Y	N	5.17.2.12
14h	O[6]	Primary Controller Capabilities data structure for the specified primary controller.	N	Y	N	5.17.2.13

Figure 273: Identify – CNS Values

CNS Value	O/M[1]	Definition	NSID[2]	CNTID[3]	CSI[4]	Reference Section
15h	O[6]	Secondary Controller list of controllers associated with the primary controller processing the command.	N	Y	N	5.17.2.14
16h	O[11]	A Namespace Granularity List (refer to the NVM Command Set Specification) is returned to the host for up to sixteen Namespace Granularity Entries.	N	N	N[8]	5.17.2.15
17h	O	A UUID List (refer to Figure 284) is returned to the host.	N	N	N	5.17.2.16
18h	O[10]	Domain List	N	N	N	5.17.2.17
19h	O[9]	Endurance Group List	N	N	N	5.17.2.18
1Ah	O[5]	I/O Command Set specific Allocated Namespace ID list	Y	N	Y	5.17.2.19
1Bh	O[5]	I/O Command Set specific Identify Namespace data structure.	Y	N	Y	5.17.2.20
1Ch	O	I/O Command Set data structure	N	Y	N	5.17.2.21
18h to 1Fh		Reserved				
Future Definition						
20h to FFh		Reserved				

Notes:
1. O/M definition: O = Optional, M = Mandatory.
2. The NSID field is used: Y = Yes, N = No.
3. The CDW10.CNTID field is used: Y = Yes, N = No.
4. The CDW11.CSI field is used: Y = Yes, N = No.
5. Mandatory for controllers that support the Namespace Management capability (refer to section 8.11).
6. Mandatory for controllers that support Virtualization Enhancements (refer to section 8.26).
7. Selection of a UUID may be supported (refer to section 8.25).
8. This Identify data structure applies to namespaces that are associated with command sets that specify logical blocks (e.g., Command Set Identifier 0h or Command Set Identifier 2h).
9. Mandatory for controllers that support Variable Capacity Management (refer to section 8.3.3).
10. Mandatory for controllers that support Capacity Management (refer to section 8.3) in an NVM subsystem that supports multiple domains (refer to section 3.2.4).
11. Only applicable for the NVM Command Set and I/O Command Sets based on the NVM Command Set. Prohibited for all other I/O Command Sets.

The Command Set Identifier values are defined in Figure 274.

Figure 274: Command Set Identifiers

Command Set Identifier Value	Description	Reference Section
00h	NVM Command Set	Refer to the NVM Command Set Specification
01h	Key Value Command Set	Refer to the Key Value Command Set Specification
02h	Zoned Namespace Command Set	Refer to the Zoned Namespace Command Set Specification
03h to 2Fh	Reserved	
30h to 3Fh	Vendor specific	
40h to FFh	Reserved	

5.17.2 Identify Data Structures

5.17.2.1 Identify Controller Data Structure (CNS 01h)

The Identify Controller data structure (refer to Figure 275) is returned to the host for the controller processing the command.

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

Bytes	I/O[1]	Admin[1]	Disc[1]	Description
Controller Capabilities and Features				
01:00	M	M	R	**PCI Vendor ID (VID):** Contains the company vendor identifier that is assigned by the PCI SIG. This is the same value as reported in the ID register in the PCI Header section of the NVMe over PCIe Transport Specification.
03:02	M	M	R	**PCI Subsystem Vendor ID (SSVID):** Contains the company vendor identifier that is assigned by the PCI SIG for the subsystem. This is the same value as reported in the SS register in the PCI Header section of the NVMe over PCIe Transport Specification.
23:04	M	M	R	**Serial Number (SN):** Contains the serial number for the NVM subsystem that is assigned by the vendor as an ASCII string. Refer to section 4.5.1 for unique identifier requirements. Refer to section 1.4.2 for ASCII string requirements.
63:24	M	M	R	**Model Number (MN):** Contains the model number for the NVM subsystem that is assigned by the vendor as an ASCII string. Refer to section 4.5.1 for unique identifier requirements. Refer to section 1.4.2 for ASCII string requirements.
71:64	M	M	M	**Firmware Revision (FR):** Contains the currently active firmware revision, as an ASCII string, for the domain of which this controller is a part. This is the same revision information that may be retrieved with the Get Log Page command, refer to section 5.16.1.4.
72	M	M	R	**Recommended Arbitration Burst (RAB):** This is the recommended Arbitration Burst size. The value is in commands and is reported as a power of two (2^*n*). This is the same units as the Arbitration Burst size. Refer to section 3.4.4.
75:73	M	M	R	**IEEE OUI Identifier (IEEE):** Contains the Organization Unique Identifier (OUI) for the controller vendor. The OUI shall be a valid IEEE/RAC assigned identifier that may be registered at http://standards.ieee.org/develop/regauth/oui/public.html.
76	O	O	R	**Controller Multi-Path I/O and Namespace Sharing Capabilities (CMIC):** This field specifies multi-path I/O and namespace sharing capabilities of the controller and NVM subsystem. Bits 7:4 are reserved. Bit 3 if set to '1', then the NVM subsystem supports Asymmetric Namespace Access Reporting (refer to section 8.1). If cleared to '0', then the NVM subsystem does not support Asymmetric Namespace Access Reporting. Bit 2 if set to '1', then the controller is associated with an SR-IOV Virtual Function. If cleared to '0', then the controller is associated with a PCI Function or a Fabrics connection. Bit 1 if set to '1', then the NVM subsystem may contain two or more controllers. If cleared to '0', then the NVM subsystem contains only a single controller. As described in section 2.4.1, an NVM subsystem that contains multiple controllers may be used by multiple hosts, or may provide multiple paths for a single host. Bit 0 if set to '1', then the NVM subsystem may contain more than one NVM subsystem port. If cleared to '0', then the NVM subsystem contains only a single NVM subsystem port.

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

Bytes	I/O[1]	Admin[1]	Disc[1]	Description
77	M	M	M	**Maximum Data Transfer Size (MDTS):** This field indicates the maximum data transfer size for a command that transfers data between host-accessible memory (refer to section 1.5.26) and the controller. The host should not submit a command that exceeds this maximum data transfer size. If a command is submitted that exceeds this transfer size, then the command is aborted with a status code of Invalid Field in Command. The value is in units of the minimum memory page size (CAP.MPSMIN) and is reported as a power of two (2^*n*). A value of 0h indicates that there is no maximum data transfer size. This field includes the length of metadata, if metadata is interleaved with the user data. This field does not apply to commands that do not transfer data between host-accessible memory and the controller (e.g., the Verify command, the Write Uncorrectable command, and the Write Zeroes command); refer to the ONCS field for restrictions on these commands and other commands that transfer data. If SGL Bit Bucket descriptors are supported, their lengths shall be included in determining if a command exceeds the Maximum Data Transfer Size for destination data buffers. Their length in a source data buffer is not included for a Maximum Data Transfer Size calculation.
79:78	M	M	M	**Controller ID (CNTLID):** Contains the NVM subsystem unique controller identifier associated with the controller.
83:80	M	M	M	**Version (VER):** This field contains the value reported in the Version property (i.e., VS property) defined in section 3.1.3.2. Implementations compliant with NVM Express Base Specification, Revision 1.2 or later shall report a non-zero value in this field.
87:84	M	M	R	**RTD3 Resume Latency (RTD3R):** This field indicates the expected latency in microseconds to resume from Runtime D3 (RTD3). Refer to section 8.15.4. A value of 0h indicates RTD3 Resume Latency is not reported.
91:88	M	M	R	**RTD3 Entry Latency (RTD3E):** This field indicates the typical latency in microseconds to enter Runtime D3 (RTD3). Refer to section 8.15.4. A value of 0h indicates RTD3 Entry Latency is not reported.

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

Bytes	I/O[1]	Admin[1]	Disc[1]	Description
95:92	M	M	M	**Optional Asynchronous Events Supported (OAES):** This field indicates the optional asynchronous events supported by the controller. A controller shall not send optional asynchronous events before they are enabled by host software. Bit 31 is set to '1' if the controller supports sending Discovery Log Page Change Notifications. If cleared to '0', then the controller does not support the Discovery Log Page Change Notification events. Bits 30:28 are reserved. Bit 27 is set to '1' if the controller supports the Zone Descriptor Changed Notices event and the associated Changed Zone List log page (refer to the Zoned Namespace Command Set Specification). If cleared to '0', then the controller does not support the Zone Descriptor Changed Notices event nor the associated Changed Zone List log page. Bits 26:16 are reserved. Bit 15 is set to '1' if the controller supports the Normal NVM Subsystem Shutdown event. If cleared to '0', then the controller does not support the Normal NVM Subsystem Shutdown event. Bit 14 is set to '1' if the controller supports the Endurance Group Event Aggregate Log Page Change Notices event. If cleared to '0', then the controller does not support the Endurance Group Event Aggregate Log Page Change Notices event. Bit 13 is set to '1' if the controller supports the LBA Status Information Alert Notices event (refer to the NVM Command Set Specification). If cleared to '0', then the controller does not support the LBA Status Information Alert Notices event. Bit 12 is set to '1' if the controller supports the Predictable Latency Event Aggregate Log Change Notices event. If cleared to '0', then the controller does not support the Predictable Latency Event Aggregate Log Change Notices event. Bit 11 is set to '1' if the controller supports sending Asymmetric Namespace Access Change Notices. If cleared to '0', then the controller does not support the Asymmetric Namespace Access Change Notices event. Bit 10 is reserved. Bit 9 is set to '1' if the controller supports the Firmware Activation Notices event. If cleared to '0', then the controller does not support the Firmware Activation Notices event. Bit 8 is set to '1' if the controller supports the Namespace Attribute Notices event and the associated Changed Namespace List log page. If cleared to '0', then the controller does not support the Namespace Attribute Notices event nor the associated Changed Namespace List log page. Bits 7:0 are reserved.

<table>
<tr>
<td>99:96</td>
<td>M</td>
<td>M</td>
<td>R</td>
<td>
Controller Attributes (CTRATT): This field indicates attributes of the controller.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:16</td><td>Reserved</td></tr>
<tr><td>15</td><td>Extended LBA Formats Supported (ELBAS): If set to ‘1’ indicates that the controller supports the I/O command set specific extended protection information formats (refer to the Protection Information Formats section of the applicable I/O command set specification).

If cleared to ‘0’ indicates that the controller does not support the I/O command set specific extended protection information formats (refer to the Protection Information Formats section of the NVM Command Set Specification).

Refer to the LBA Format Extension Enable (LBAFEE) field in the Host Behavior Support feature (refer to section 5.27.1.18) for details for host software to enable the controller to operate on namespaces using the protection information formats.

NOTE: This bit field applies to all I/O Command Sets. The original name has been retained for historical continuity.</td></tr>
<tr><td>14</td><td>Delete NVM Set: If set to ‘1’, then the controller supports the Delete NVM Set operation (refer to section 8.3.3). If cleared to ‘0’, then the controller does not support the Delete NVM Set operation.</td></tr>
<tr><td>13</td><td>Delete Endurance Group: If set to ‘1’, then the controller supports the Delete Endurance Group operation (refer to section 8.3.3). If cleared to ‘0’, then the controller does not support the Delete Endurance Group operation.</td></tr>
<tr><td>12</td><td>Variable Capacity Management: If set to ‘1’, then the controller supports Variable Capacity Management (refer to section 8.3.3). If cleared to ‘0’, then the controller does not support Variable Capacity Management.</td></tr>
<tr><td>11</td><td>Fixed Capacity Management: If set to ‘1’, then the controller supports Fixed Capacity Management (refer to section 8.3.2). If cleared to ‘0’, then the controller does not support Fixed Capacity Management.</td></tr>
<tr><td>10</td><td>Multi-Domain Subsystem (MDS): If set to ‘1’, then the NVM subsystem supports the multiple domains (refer to section 3.2.4). If cleared to ‘0’, then the NVM subsystem does not support the reporting of multiple domains and the NVM subsystem consists of a single domain.</td></tr>
<tr><td>9</td><td>UUID List: If set to ‘1’, then the controller supports reporting of a UUID List (refer to Figure 284). If cleared to ‘0’, then the controller does not support reporting of a UUID List (refer to section 8.25).</td></tr>
<tr><td>8</td><td>SQ Associations: If set to ‘1’, then the controller supports SQ Associations (refer to section 8.22). If cleared to ‘0’, then the controller does not support SQ Associations.</td></tr>
<tr><td>7</td><td>Namespace Granularity: If set to ‘1’, then the controller supports reporting of Namespace Granularity (refer to section 5.17.2.15). If cleared to ‘0’, the controller does not support reporting of Namespace Granularity. If the Namespace Management capability (refer to section 8.11) is not supported, then this bit shall be cleared to ‘0’.</td></tr>
<tr><td>6</td><td>Traffic Based Keep Alive Support (TBKAS): If set to '1‘, then the controller supports restarting the Keep Alive Timer if an Admin command or an I/O command is processed during the Keep Alive Timeout Interval (refer to section 3.9.2). If cleared to '0‘, then the controller supports restarting the Keep Alive Timer only if a Keep Alive command is processed during the Keep Alive Timeout Interval (refer to section 3.9.1).</td></tr>
<tr><td>5</td><td>Predictable Latency Mode: If set to ‘1’, then the controller supports Predictable Latency Mode (refer to section 8.16). If cleared to ‘0’, then the controller does not support Predictable Latency Mode.</td></tr>
<tr><td>4</td><td>Endurance Groups: If set to ‘1’, then the controller supports Endurance Groups (refer to section 3.2.3). If cleared to ‘0’, then the controller does not support Endurance Groups.</td></tr>
</table>
</td>
</tr>
</table>

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

<table>
<tr><th>Bytes</th><th>I/O[1]</th><th>Admin[1]</th><th>Disc[1]</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td></td><td>
<table>
<tr><td>3</td><td>Read Recovery Levels: If set to ‘1’, then the controller supports Read Recovery Levels (refer to section 8.17). If cleared to ‘0’, then the controller does not support Read Recovery Levels.</td></tr>
<tr><td>2</td><td>NVM Sets: If set to ‘1’, then the controller supports NVM Sets (refer to section 3.2.2). If cleared to ‘0’, then the controller does not support NVM Sets.</td></tr>
<tr><td>1</td><td>Non-Operational Power State Permissive Mode: If set to ‘1’, then the controller supports host control of whether the controller may temporarily exceed the power of a non-operational power state for the purpose of executing controller initiated background operations in a non-operational power state (i.e., Non-Operational Power State Permissive Mode supported). If cleared to ‘0’, then the controller does not support host control of whether the controller may exceed the power of a non-operational state for the purpose of executing controller initiated background operations in a non-operational state (i.e., Non-Operational Power State Permissive Mode not supported). Refer to section 5.27.1.14.</td></tr>
<tr><td>0</td><td>Host Identifier Support: If set to ‘1’, then the controller supports a 128-bit Host Identifier. Bit 0 if cleared to ‘0’, then the controller does not support a 128-bit Host Identifier.</td></tr>
</table>
</td></tr>
<tr><td>101:100</td><td>O</td><td>O</td><td>R</td><td>
Read Recovery Levels Supported (RRLS): If Read Recovery Levels (RRL) are supported, then this field shall be supported. If a bit is set to ‘1’, then the corresponding Read Recovery Level is supported. If a bit is cleared to ‘0’, then the corresponding Read Recovery Level is not supported.
<table>
<tr><th>Bit</th><th>Definition</th></tr>
<tr><td>0</td><td>Read Recovery Level 0</td></tr>
<tr><td>1</td><td>Read Recovery Level 1</td></tr>
<tr><td>2</td><td>Read Recovery Level 2</td></tr>
<tr><td>3</td><td>Read Recovery Level 3</td></tr>
<tr><td>4</td><td>Read Recovery Level 4 – Default[1]</td></tr>
<tr><td>5</td><td>Read Recovery Level 5</td></tr>
<tr><td>6</td><td>Read Recovery Level 6</td></tr>
<tr><td>7</td><td>Read Recovery Level 7</td></tr>
<tr><td>8</td><td>Read Recovery Level 8</td></tr>
<tr><td>9</td><td>Read Recovery Level 9</td></tr>
<tr><td>10</td><td>Read Recovery Level 10</td></tr>
<tr><td>11</td><td>Read Recovery Level 11</td></tr>
<tr><td>12</td><td>Read Recovery Level 12</td></tr>
<tr><td>13</td><td>Read Recovery Level 13</td></tr>
<tr><td>14</td><td>Read Recovery Level 14</td></tr>
<tr><td>15</td><td>Read Recovery Level 15 – Fast Fail[1]</td></tr>
<tr><td colspan="2">NOTE:
1. If Read Recovery Levels are supported, then this bit shall be set to ‘1’.</td></tr>
</table>
</td></tr>
<tr><td>110:102</td><td></td><td></td><td></td><td>Reserved</td></tr>
</table>

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

<table>
<tr><th>Bytes</th><th>I/O[1]</th><th>Admin[1]</th><th>Disc[1]</th><th>Description</th></tr>
<tr><td>111</td><td>M</td><td>M</td><td>M</td><td>Controller Type (CNTRLTYPE): This field specifies the controller type. A value of 0h indicates that the controller type is not reported.

Implementations compliant with NVM Express Base Specification, Revision 1.4 or later shall report a controller type (i.e., the value 0h is reserved and shall not be used). Implementations compliant with an earlier specification version may report a value of 0h to indicate that a controller type is not reported.

<table>
<tr><th>Value</th><th>Controller Type</th></tr>
<tr><td>0h</td><td>Reserved (controller type not reported)</td></tr>
<tr><td>1h</td><td>I/O controller</td></tr>
<tr><td>2h</td><td>Discovery controller</td></tr>
<tr><td>3h</td><td>Administrative controller</td></tr>
<tr><td>4h to FFh</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>127:112</td><td>O</td><td>O</td><td>R</td><td>FRU Globally Unique Identifier (FGUID): This field contains a 128-bit value that is globally unique for a given Field Replaceable Unit (FRU). Refer to the NVM Express® Management Interface Specification for the definition of a FRU. This field remains fixed throughout the life of the FRU. This field shall contain the same value for each controller associated with a given FRU.

This field uses the EUI-64 based 16-byte designator format. Bytes 122:120 contain the 24-bit Organizationally Unique Identifier (OUI) value assigned by the IEEE Registration Authority. Bytes 127:123 contain an extension identifier assigned by the corresponding organization. Bytes 119:112 contain the vendor specific extension identifier assigned by the corresponding organization. Refer to the IEEE EUI-64 guidelines for more information. This field is big endian (refer to section 4.3.4).

When not implemented, this field contains a value of 0h.</td></tr>
<tr><td>129:128</td><td>O</td><td>O</td><td>R</td><td>Command Retry Delay Time 1 (CRDT1): If the Do Not Retry (DNR) bit is cleared to ‘0’ in the CQE and the Command Retry Delay (CRD) field is set to 01b in the CQE, then this value indicates the command retry delay time in units of 100 milliseconds.</td></tr>
<tr><td>131:130</td><td>O</td><td>O</td><td>R</td><td>Command Retry Delay Time 2 (CRDT2): If the DNR bit is cleared to ‘0’ in the CQE and the CRD field is set to 10b in the CQE, then this value indicates the command retry delay time in units of 100 milliseconds.</td></tr>
<tr><td>133:132</td><td>O</td><td>O</td><td>R</td><td>Command Retry Delay Time 3 (CRDT3): If the DNR bit is cleared to ‘0’ in the CQE and CRD field is set to 11b in the CQE, then this value indicates the command retry delay time in units of 100 milliseconds.</td></tr>
<tr><td>239:134</td><td></td><td></td><td></td><td>Reserved</td></tr>
<tr><td>252:240</td><td></td><td></td><td></td><td>Reserved for the NVMe Management Interface.</td></tr>
<tr><td>253</td><td>M</td><td>M</td><td>M</td><td>NVM Subsystem Report (NVMSR): This field reports information associated with the NVM subsystem. If the controller is compliant with the NVM Express Management Interface Specification, then at least one bit in this field is set to ‘1’. If the NVM subsystem does not support the NVM Express Management Interface Specification, then this field shall be cleared to 0h. Refer to the NVM Express Management Interface Specification.

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:2</td><td>Reserved</td></tr>
<tr><td>1</td><td>NVMe Enclosure (NVMEE): If set to ‘1’, then the NVM subsystem is part of an NVMe Enclosure. If cleared to ‘0’, then the NVM subsystem is not part of an NVMe Enclosure.</td></tr>
<tr><td>0</td><td>NVMe Storage Device (NVMESD): If set to ‘1’, then the NVM subsystem is part of an NVMe Storage Device. If cleared to ‘0’, then the NVM subsystem is not part of an NVMe Storage Device.</td></tr>
</table>
</td></tr>
</table>

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

<table>
<tr><th>Bytes</th><th>I/O[1]</th><th>Admin[1]</th><th>Disc[1]</th><th>Description</th></tr>
<tr><td>254</td><td>M</td><td>M</td><td>M</td><td>VPD Write Cycle Information (VWCI): This field indicates information about the remaining number of times that VPD contents are able to be updated using the VPD Write command. Refer to the NVM Express Management Interface Specification for details on VPD contents and the VPD Write command.

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7</td><td>VPD Write Cycles Remaining Valid (VWCRV): If this bit is set to ‘1’, then the VPD Write Cycles Remaining field is valid. If this bit is cleared to ‘0’, then the VPD Write Cycles Remaining field is invalid and cleared to ‘0’.</td></tr>
<tr><td>6:0</td><td>VPD Write Cycles Remaining (VWCR): If the VPD Write Cycle Remaining Valid bit is set to ‘1’, then this field contains a value indicating the remaining number of times that VPD contents are able to be updated in units of 256 bytes using the VPD Write command. For example, a 1 KiB FRU Information Device that can be updated 8 times would indicate a value of 32 in this field. If this field is set to 7Fh, then the remaining number of times that VPD contents are able to be updated using the VPD Write command is greater than or equal to 7Fh.
If the VPD Write Cycle Remaining Valid bit is cleared to ‘0’, then this field is not valid and shall be cleared to a value of 0h.</td></tr>
</table></td></tr>
<tr><td>255</td><td>M</td><td>M</td><td>M</td><td>Management Endpoint Capabilities (MEC): This field indicates the capabilities of the Management Endpoint in the NVM subsystem. Refer to the NVM Express Management Interface Specification for details.

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

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

Bytes	I/O[1]	Admin[1]	Disc[1]	Description
Admin Command Set Attributes & Optional Controller Capabilities				
257:256	M	M	R	**Optional Admin Command Support (OACS):** This field indicates the optional Admin commands and features supported by the controller. Refer to section 3.1.2. Bits 15:11 are reserved. Bit 10 if set to '1', then the controller supports the Command and Feature Lockdown capability (refer to section 8.4). If cleared to '0', then the controller does not support the Command and Feature Lockdown capability. This value shall be the same for all controllers in the NVM subsystem. Bit 9 if set to '1', then the controller supports the Get LBA Status capability (refer to the NVM Command Set Specification). If cleared to '0', then the controller does not support the Get LBA Status capability. Bit 8 if set to '1', then the controller supports the Doorbell Buffer Config command. If cleared to '0', then the controller does not support the Doorbell Buffer Config command. Bit 7 if set to '1', then the controller supports the Virtualization Management command. If cleared to '0', then the controller does not support the Virtualization Management command. Bit 6 if set to '1', then the controller supports the NVMe-MI Send and NVMe-MI Receive commands. If cleared to '0', then the controller does not support the NVMe-MI Send and NVMe-MI Receive commands. Bit 5 if set to '1', then the controller supports Directives. If cleared to '0', then the controller does not support Directives. A controller that supports Directives shall support the Directive Send and Directive Receive commands. Refer to section 8.7. Bit 4 if set to '1', then the controller supports the Device Self-test command. If cleared to '0', then the controller does not support the Device Self-test command. Bit 3 if set to '1', then the controller supports the Namespace Management capability (refer to section 8.11). If cleared to '0', then the controller does not support the Namespace Management capability. Bit 2 if set to '1', then the controller supports the Firmware Commit and Firmware Image Download commands. If cleared to '0', then the controller does not support the Firmware Commit and Firmware Image Download commands. Bit 1 if set to '1', then the controller supports the Format NVM command. If cleared to '0', then the controller does not support the Format NVM command. Bit 0 if set to '1', then the controller supports the Security Send and Security Receive commands. If cleared to '0', then the controller does not support the Security Send and Security Receive commands.
258	M	M	R	**Abort Command Limit (ACL):** This field is used to convey the maximum number of concurrently executing Abort commands supported by the controller (refer to section 5.1). This is a 0's based value. It is recommended that implementations support concurrent execution of a minimum of four Abort commands.
259	M	M	R	**Asynchronous Event Request Limit (AERL):** This field is used to convey the maximum number of concurrently outstanding Asynchronous Event Request commands supported by the controller (refer to section 5.2). This is a 0's based value. It is recommended that implementations support a minimum of four Asynchronous Event Request Limit commands outstanding simultaneously.

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

<table>
<tr><th>Bytes</th><th>I/O[1]</th><th>Admin[1]</th><th>Disc[1]</th><th>Description</th></tr>
<tr><td>260</td><td>M</td><td>M</td><td>R</td><td>Firmware Updates (FRMW): This field indicates capabilities regarding firmware updates. Refer to section 3.11 for more information on the firmware update process.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:6</td><td>Reserved</td></tr>
<tr><td>5</td><td>Support Multiple Update Detection (SMUD): If set to ‘1’ indicates that the controller is able to detect overlapping firmware/boot partition image update command sequences (refer to section 3.11 and section 8.2.2). If cleared to ‘0’, then the controller is not able to detect overlapping firmware/boot partition image update command sequences.</td></tr>
<tr><td>4</td><td>Firmware Activation Without Reset (FAWR): If set to ‘1’ indicates that the controller supports firmware activation without a reset. If cleared to ‘0’, then the controller requires a reset for firmware to be activated.</td></tr>
<tr><td>3:1</td><td>Number Of Firmware Slots (NOFS): This field indicates the number of firmware slots supported by the domain that contains this controller. This field shall specify a value from one to seven, indicating that at least one firmware slot is supported and up to seven maximum. This corresponds to firmware slots 1 through 7</td></tr>
<tr><td>0</td><td>First Firmware Slot Read Only (FFSRO): If set to ‘1’ indicates that the first firmware slot (i.e., slot 1) is read only. If cleared to ‘0’, then the first firmware slot (i.e., slot 1) is read/write. Implementations may choose to have a baseline read only firmware image.</td></tr>
</table></td></tr>
</table>

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

<table>
<tr><th>Bytes</th><th>I/O[1]</th><th>Admin[1]</th><th>Disc[1]</th><th>Description</th></tr>
<tr><td>261</td><td>M</td><td>M</td><td>M</td><td>Log Page Attributes (LPA): This field indicates optional attributes for log pages that are accessed via the Get Log Page command.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7</td><td>Reserved</td></tr>
<tr><td>6</td><td>If set to ‘1’, then the controller supports Data Area 4 for the Telemetry Host-Initiated and Telemetry Controller-Initiated log. If cleared to ’0’, then the controller does not support Data Area 4 for the Telemetry Host-Initiated and Telemetry Controller-Initiated log pages.</td></tr>
<tr><td>5</td><td>If set to ‘1’, then the controller supports:
the Supported Log Pages log page (Log Identifier 0h);
returning the scope of each command in the Commands Supported and Effects log page (Log Identifier 05h);
the Feature Identifiers Supported and Effects log page (Log Identifier 12h); and
the NVMe-MI Commands Supported and Effects log page (Log Identifier 13h).
If cleared to ‘0’, then the controller:
does not support returning the scope of each command in the Commands Supported and Effects log page;
may support the Supported Log Pages log page;
may support the Feature Identifiers Supported and Effects log page; and
may support the NVMe-MI Commands Supported and Effects log page.</td></tr>
<tr><td>4</td><td>If set to ‘1’, then the controller supports the Persistent Event log. If cleared to ‘0’, then the controller does not support the Persistent Event log.</td></tr>
<tr><td>3</td><td>If set to ‘1’, then the controller supports the Telemetry Host-Initiated and Telemetry Controller-Initiated log pages and sending Telemetry Log Notices. If cleared to ’0’, then the controller does not support the Telemetry Host-Initiated and Telemetry Controller-Initiated log pages and Telemetry Log Notice events.</td></tr>
<tr><td>2</td><td>If set to ‘1’, then the controller supports extended data for the Get Log Page command (including extended Number of Dwords and Log Page Offset fields). If cleared to ‘0’, then the controller does not support extended data for the Get Log Page command.</td></tr>
<tr><td>1</td><td>If set to ‘1’, then the controller supports the Commands Supported and Effects log page. Bit 1 if cleared to ‘0’, then the controller does not support the Commands Supported and Effects log page.</td></tr>
<tr><td>0</td><td>If set to ‘1’, then the controller supports the SMART / Health Information log page on a per namespace basis. If cleared to ‘0’, then the controller does not support the SMART / Health Information log page on a per namespace basis.</td></tr>
</table></td></tr>
<tr><td>262</td><td>M</td><td>M</td><td>M</td><td>Error Log Page Entries (ELPE): This field indicates the maximum number of Error Information Log Entries that are stored by the controller. This field is a 0’s based value.</td></tr>
<tr><td>263</td><td>M</td><td>M</td><td>R</td><td>Number of Power States Support (NPSS): This field indicates the number of NVM Express power states supported by the controller. This is a 0’s based value. Refer to section 8.15.

Power states are numbered sequentially starting at power state 0. A controller shall support at least one power state (i.e., power state 0) and may support up to 31 additional power states (i.e., up to 32 total).</td></tr>
</table>

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

Bytes	I/O[1]	Admin[1]	Disc[1]	Description
264	M	M	R	**Admin Vendor Specific Command Configuration (AVSCC):** This field indicates the configuration settings for Admin Vendor Specific command handling. Refer to section 8.23. Bits 7:1 are reserved. Bit 0 if set to '1' indicates that all Admin Vendor Specific Commands use the format defined in Figure 88. If cleared to '0' indicates that the format of all Admin Vendor Specific Commands are vendor specific.
265	O	O	R	**Autonomous Power State Transition Attributes (APSTA):** This field indicates the attributes of the autonomous power state transition feature. Refer to section 8.15.2. Bits 7:1 are reserved. Bit 0 if set to '1', then the controller supports autonomous power state transitions. If cleared to '0', then the controller does not support autonomous power state transitions.
267:266	M	M	R	**Warning Composite Temperature Threshold (WCTEMP):** This field indicates the minimum Composite Temperature field value (reported in the SMART / Health Information log in Figure 207) that indicates an overheating condition during which controller operation continues. Immediate remediation is recommended (e.g., additional cooling or workload reduction). The platform should strive to maintain a composite temperature less than this value. A value of 0h in this field indicates that no warning temperature threshold value is reported by the controller. Implementations compliant with NVM Express Base Specification, Revision 1.2 or later shall report a non-zero value in this field. It is recommended that implementations report a value of 0157h in this field.
269:268	M	M	R	**Critical Composite Temperature Threshold (CCTEMP):** This field indicates the minimum Composite Temperature field value (reported in the SMART / Health Information log in Figure 207) that indicates a critical overheating condition (e.g., may prevent continued normal operation, possibility of data loss, automatic device shutdown, extreme performance throttling, or permanent damage). A value of 0h in this field indicates that no critical temperature threshold value is reported by the controller. Implementations compliant with NVM Express Base Specification, Revision 1.2 or later shall report a non-zero value in this field.
271:270	O	O	R	**Maximum Time for Firmware Activation (MTFA):** Indicates the maximum time the controller temporarily stops processing commands to activate the firmware image. This field shall be valid if the controller supports firmware activation without a reset. This field is specified in 100 millisecond units. A value of 0h indicates that the maximum time is undefined.
275:272	O	O	R	**Host Memory Buffer Preferred Size (HMPRE):** This field indicates the preferred size that the host is requested to allocate for the Host Memory Buffer feature in 4 KiB units. This value shall be greater than or equal to the Host Memory Buffer Minimum Size. If this field is non-zero, then the Host Memory Buffer feature is supported. If this field is cleared to 0h, then the Host Memory Buffer feature is not supported.
279:276	O	O	R	**Host Memory Buffer Minimum Size (HMMIN):** This field indicates the minimum size that the host is requested to allocate for the Host Memory Buffer feature in 4 KiB units. If this field is cleared to 0h, then the host is requested to allocate any amount of host memory possible up to the HMPRE value.
295:280	O	O	R	**Total NVM Capacity (TNVMCAP):** This field indicates the total NVM capacity that is accessible by the controller. The value is in bytes. This field shall be supported if the Namespace Management capability (refer to section 8.11) is supported or if the Capacity Management capability (refer to section 8.3) is supported. Refer to section 3.8.

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

<table>
<tr><th>Bytes</th><th>I/O[1]</th><th>Admin[1]</th><th>Disc[1]</th><th>Description</th></tr>
<tr><td>311:296</td><td>O</td><td>O</td><td>R</td><td>Unallocated NVM Capacity (UNVMCAP): This field indicates the unallocated NVM capacity that is accessible by the controller. The value is in bytes. This field shall be supported if the Namespace Management capability (refer to section 8.11) is supported or if the Capacity Management capability (refer to section 8.3) is supported.

Refer to section 3.8.</td></tr>
<tr><td>315:312</td><td>O</td><td>O</td><td>R</td><td>Replay Protected Memory Block Support (RPMBS): This field indicates if the controller supports one or more Replay Protected Memory Blocks (RPMBs) and the capabilities. Refer to section 8.18.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:24</td><td>Access Size: If the Number of RPMB Units field is non-zero, then this field indicates the maximum number of 512B units of data that may be read or written per RPMB access by Security Send or Security Receive commands for the controller. This is a 0’s based value. A value of 0h indicates support for one unit of 512B of data.

If the Number of RPMB Units field is 0h, then this field shall be ignored.</td></tr>
<tr><td>23:16</td><td>Total Size: If the Number of RPMB Units field is non-zero, then this field indicates the number of 128 KiB units of data in each RPMB supported in the controller. This is a 0’s based value. A value of 0h indicates support for one unit of 128 KiB of data.

If the Number of RPMB Units field is 0h, this field shall be ignored.</td></tr>
<tr><td>15:06</td><td>Reserved</td></tr>
<tr><td>05:03</td><td>Authentication Method: This field indicates the authentication method used to access all RPMBs in the controller. The values for this field are:
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>HMAC SHA-256 (refer to RFC 6234)</td></tr>
<tr><td>001b to 111b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>02:00</td><td>Number of RPMB Units: This field indicates the number of RPMB targets the controller supports. All RPMB targets supported shall have the same capabilities as defined in the RPMBS field. A value of 0h indicates the controller does not support Replay Protected Memory Blocks. If this value is non-zero, then the controller shall support the Security Send and Security Receive commands.</td></tr>
</table></td></tr>
<tr><td>317:316</td><td>O</td><td>O</td><td>R</td><td>Extended Device Self-test Time (EDSTT): If the Device Self-test command is supported, then this field indicates the nominal amount of time in one minute units that the controller takes to complete an extended device self-test operation when in power state 0. If the Device Self-test command is not supported, then this field is reserved.</td></tr>
<tr><td>318</td><td>O</td><td>O</td><td>R</td><td>Device Self-test Options (DSTO): This field indicates the optional Device Self-test command or operation behaviors supported by the controller or NVM subsystem.
Bits 7:1 are reserved.

Bit 0 if set to ‘1’, then the NVM subsystem supports only one device self-test operation in progress at a time. If cleared to ‘0’, then the NVM subsystem supports one device self-test operation per controller at a time.</td></tr>
</table>

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

Bytes	I/O[1]	Admin[1]	Disc[1]	Description
319	M	M	R	**Firmware Update Granularity (FWUG):** This field indicates the granularity and alignment requirement of the firmware image being updated by the Firmware Image Download command (refer to section 5.13). If the values specified in the NUMD field or the OFST field in the Firmware Image Download command do not conform to this granularity and alignment requirement, then the firmware update may abort with a status code of Invalid Field in Command. For the broadest interoperability with host software, it is recommended that the controller set this value to the lowest value possible. The value is reported in 4 KiB units (e.g., 1h corresponds to 4 KiB, 2h corresponds to 8 KiB). A value of 0h indicates that no information on granularity is provided. A value of FFh indicates there is no restriction (i.e., any granularity and alignment in dwords is allowed).
321:320	M	M	R	**Keep Alive Support (KAS):** This field indicates the granularity of the Keep Alive Timer in 100 millisecond units (refer to section 3.9). If this field is cleared to 0h, then the Keep Alive feature is not supported. The Keep Alive feature shall be supported for NVMe over Fabrics implementations as described in section 3.9.
323:322	O	O	R	**Host Controlled Thermal Management Attributes (HCTMA):** This field indicates the attributes of the host controlled thermal management feature. Refer to section 8.15.5. Bits 15:1 are reserved. Bit 0 if set to '1', then the controller supports host controlled thermal management. If cleared to '0', then the controller does not support host controlled thermal management. If this bit is set to '1', then the controller shall support the Set Features command and Get Features command with the Feature Identifier field set to 10h.
325:324	O	O	R	**Minimum Thermal Management Temperature (MNTMT):** This field indicates the minimum temperature, in Kelvins, that the host may request in the Thermal Management Temperature 1 field and Thermal Management Temperature 2 field of a Set Features command with the Feature Identifier field set to 10h. A value of 0h indicates that the controller does not report this field or the host controlled thermal management feature (refer to section 8.15.5) is not supported.
327:326	O	O	R	**Maximum Thermal Management Temperature (MXTMT):** This field indicates the maximum temperature, in Kelvins, that the host may request in the Thermal Management Temperature 1 field and Thermal Management Temperature 2 field of the Set Features command with the Feature Identifier set to 10h. A value of 0h indicates that the controller does not report this field or the host controlled thermal management feature is not supported.

<table>
<tr>
<td>331:328</td>
<td>O</td>
<td>O</td>
<td>R</td>
<td>
Sanitize Capabilities (SANICAP): This field indicates attributes for sanitize operations. If the Sanitize command is supported, then this field shall be non-zero. If the Sanitize command is not supported, then this field shall be cleared to 0h. Refer to section 8.21.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:30</td><td>
No-Deallocate Modifies Media After Sanitize (NODMMAS): This field indicates if media is additionally modified by the controller after a sanitize operation successfully completes that had been started by a Sanitize command with the No-Deallocate After Sanitize bit set to ‘1’.

The work required for the associated additional media modification is included both in the estimated time for each sanitize operation and in the Sanitize Progress field (refer to Figure 267).
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>Additional media modification after sanitize operation completes successfully is not defined. Only controllers compliant with NVM Express Base Specification, Revision 1.3 and earlier or that have bits 2:0 of the SANICAP field cleared to 000b are allowed to return this value.</td></tr>
<tr><td>01b</td><td>Media is not additionally modified by the controller after sanitize operation completes successfully.</td></tr>
<tr><td>10b</td><td>Media is additionally modified by the controller after sanitize operation completes successfully. The Sanitize Operation Completed event does not occur until the additional media modification associated with this field has completed.</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table>
If bits 2:0 of the SANICAP field are cleared to 000b, then the controller shall clear this field to 00b.
</td></tr>
<tr><td>29</td><td>
No-Deallocate Inhibited (NDI): If set to ‘1’ and the No-Deallocate Response Mode bit is set to ‘1’, then the controller deallocates after the sanitize operation even if the No-Deallocate After Sanitize bit is set to ‘1’ in a Sanitize command.

If:
a) this bit is set to ‘1’;
b) the No-Deallocate After Sanitize bit is set to ‘1’ in a Sanitize command, and:
1) the No-Deallocate Response Mode bit (refer to Figure 352) is cleared to ‘0’; or
2) the Sanitize Config Feature (refer to section 5.27.1.19) is not supported,

then the controller aborts the Sanitize command with a status code of Invalid Field in Command.

If the No-Deallocate After Sanitize bit is cleared to ‘0’ in a Sanitize command, then the value of this bit has no effect on the processing that Sanitize command.

If this bit is cleared to ‘0’, then the controller supports the No-Deallocate After Sanitize bit in a Sanitize command.

If bits 2:0 of the SANICAP field are cleared to 0h, then the controller shall clear this bit to ‘0’.
</td></tr>
<tr><td>28:03</td><td>Reserved</td></tr>
</table>
</td>
</tr>
</table>

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

Bytes	I/O[1]	Admin[1]	Disc[1]	Description
				02: **Overwrite Support (OWS):** If set to '1', then the controller supports the Overwrite sanitize operation. If cleared to '0', then the controller does not support the Overwrite sanitize operation. 01: **Block Erase Support (BES):** If set to '1', then the controller supports the Block Erase sanitize operation. If cleared to '0', then the controller does not support the Block Erase sanitize operation. 00: **Crypto Erase Support (CES):** If set to '1', then the controller supports the Crypto Erase sanitize operation. If cleared to '0', then the controller does not support the Crypto Erase sanitize operation.
335:332	O	O	R	**Host Memory Buffer Minimum Descriptor Entry Size (HMMINDS):** This field indicates the minimum usable size of a Host Memory Buffer Descriptor Entry in 4 KiB units. If this field is cleared to 0h, then the controller does not indicate any limitations on the Host Memory Buffer Descriptor Entry size.
337:336	O	O	R	**Host Memory Maximum Descriptors Entries (HMMAXD**): This field indicates the number of usable Host Memory Buffer Descriptor Entries. If this field is cleared to 0h, then the controller does not indicate a maximum number of Host Memory Buffer Descriptor Entries.
339:338	O	O	R	**NVM Set Identifier Maximum (NSETIDMAX):** This field defines the maximum value of a valid NVM Set Identifier for any controller in the NVM subsystem. The number of NVM Sets supported by the NVM subsystem is less than or equal to NSETIDMAX.
341:340	O	O	R	**Endurance Group Identifier Maximum (ENDGIDMAX):** This field defines the maximum value of a valid Endurance Group Identifier for any controller in the NVM subsystem. The number of Endurance Groups supported by the NVM subsystem is less than or equal to ENDGIDMAX.
342	O	O	R	**ANA Transition Time (ANATT):** This field indicates the maximum amount of time, in seconds, for a transition between ANA states or the maximum amount of time, in seconds, that the controller reports the ANA change state. If the controller supports Asymmetric Namespace Access Reporting (refer to the CMIC field in Figure 275), then this field shall be set to a non-zero value. If the controller does not support Asymmetric Namespace Access Reporting, then this field shall be cleared to 0h. Refer to section 8.10.4.

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

Bytes	I/O[1]	Admin[1]	Disc[1]	Description
343	O	O	R	**Asymmetric Namespace Access Capabilities (ANACAP):** This field indicates the capabilities associated with Asymmetric Namespace Access Reporting (refer to section 8.1). Bit 7 if set to '1', then the controller supports a non-zero value in the ANAGRPID field of the Namespace Management command. If cleared to '0', then the controller does not support a non-zero value in the ANAGRPID field of the Namespace Management command. If the Namespace Management command is not supported, then this bit shall be cleared to '0'. Bit 6 if set to '1', then the ANAGRPID field in the Identify Namespace data structure (refer to the NVM Command Set Specification) does not change while the namespace is attached to any controller. If cleared to '0', then the ANAGRPID field may change while the namespace is attached to any controller. Refer to section 8.1.2. Bit 5 is reserved. Bit 4 if set to '1', then the controller is able to report ANA Change state (refer to section 8.1.3.5). If cleared to '0', then the controller does not report ANA Change state. Bit 3 if set to '1', then the controller is able to report ANA Persistent Loss state (refer to section 8.1.3.4). If cleared to '0', then the controller does not report ANA Persistent Loss state. Bit 2 if set to '1', then the controller is able to report ANA Inaccessible state (refer to section 8.1.3.3). If cleared to '0', then the controller does not report ANA Inaccessible state. Bit 1 if set to '1', then the controller is able to report ANA Non-Optimized state (refer to section 8.1.3.2). If cleared to '0', then the controller does not report ANA Non-Optimized state. Bit 0 if set to '1', then the controller is able to report ANA Optimized state (refer to section 8.1.3.1). If the controller supports Asymmetric Namespace Access Reporting, then this bit is set to '1'.
347:344	O	O	R	**ANA Group Identifier Maximum (ANAGRPMAX):** This field indicates the maximum value of a valid ANA Group Identifier for any controller in the NVM subsystem. If the controller supports Asymmetric Namespace Access Reporting (refer to the CMIC field in Figure 275), then this field shall be set to a non-zero value. If the controller does not support Asymmetric Namespace Access Reporting, then this field shall be cleared to 0h.
351:348	O	O	R	**Number of ANA Group Identifiers (NANAGRPID):** This field indicates the number of ANA groups supported by the controller. If the controller supports Asymmetric Namespace Access Reporting (refer to the CMIC field in Figure 275), then this field shall be set to a non-zero value that is less than or equal to the ANAGRPMAX value. If the controller does not support Asymmetric Namespace Access Reporting, then this field shall be cleared to 0h.
355:352	O	O	R	**Persistent Event Log Size (PELS):** This field indicates the maximum reportable size for the Persistent Event Log (Refer to section 5.16.1.14) in 64 KiB units. If the Persistent Event Log is not supported, then this field is reserved.
357:356	O	O	O	**Domain Identifier:** This field indicates the identifier of the domain (refer to section 3.2.4) that contains this controller. If the MDS bit is set to '1', then this field shall be set to a non-zero value. If the NVM subsystem does not support multiple domains (i.e., the NVM subsystem consists of a single domain), then this field shall be cleared to 0h.
367:358				Reserved
383:368	O	R	R	**Max Endurance Group Capacity (MEGCAP):** This field indicates the maximum capacity of a single Endurance Group. If this field is cleared to 0h, the NVM subsystem does not report a maximum Endurance Group Capacity value.
511:384				Reserved

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

Bytes	I/O[1]	Admin[1]	Disc[1]	Description
NVM Command Set Attributes				
512	M	M	R	**Submission Queue Entry Size (SQES):** This field defines the required and maximum I/O Submission Queue entry size. Bits 7:4 define the maximum I/O Submission Queue entry size when using the NVM Command Set. This value is greater than or equal to the required SQ entry size (i.e., bits 3:0 in this field). The value is in bytes and is reported as a power of two (2^*n*). The recommended value is 6, corresponding to a standard SQ entry size of 64 bytes. Controllers that implement proprietary extensions may support a larger value. Bits 3:0 define the required (i.e., minimum) I/O Submission Queue entry size. This is the minimum entry size that may be used. The value is in bytes and is reported as a power of two (2^*n*). The required value shall be 6, corresponding to 64.
513	M	M	R	**Completion Queue Entry Size (CQES):** This field defines the required and maximum I/O Completion Queue entry size. Bits 7:4 define the maximum I/O Completion Queue entry size. This value is greater than or equal to the required CQ entry size (i.e., bits 3:0 in this field). The value is in bytes and is reported as a power of two (2^*n*). The recommended value is 4, corresponding to a standard CQ entry size of 16 bytes. Controllers that implement proprietary extensions may support a larger value. Bits 3:0 define the required (i.e., minimum) I/O Completion Queue entry size. This is the minimum entry size that may be used. The value is in bytes and is reported as a power of two (2^*n*). The required value shall be 4, corresponding to 16.
515:514	M	M	M	**Maximum Outstanding Commands (MAXCMD):** Indicates the maximum number of commands that the controller processes at one time for a particular queue (which may be larger than the size of the corresponding Submission Queue). The host may use this value to size Completion Queues and optimize the number of commands submitted at one time to a particular I/O Queue. This field is mandatory for NVMe over Fabrics implementations and optional for NVMe over PCIe implementations. If the field is not used, it shall be cleared to 0h.
519:516	M	M	R	**Number of Namespaces (NN):** This field indicates the maximum value of a valid NSID for the NVM subsystem. Refer to the MNAN field for the number of supported namespaces in the NVM subsystem.

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

Bytes	I/O[1]	Admin[1]	Disc[1]	Description
521:520	M	M	R	**Optional NVM Command Support (ONCS):** This field indicates the optional I/O commands and features supported by the controller. Refer to section 3.1.2. Bits 15:9 are reserved. Bit 8 if set to '1', then the controller supports the NVM Command Set Copy command. If cleared to '0', then the controller does not support the NVM Command Set Copy command. Bit 7 if set to '1', then the controller supports the NVM Command Set Verify command and the Verify Size Limit (VSL) field indicates the recommended maximum data size for Verify commands. If cleared to '0', then controller support of the NVM Command Set Verify command is indicated by a non-zero data size limit in the VSL field. Bit 6 if set to '1', then the controller supports the Timestamp feature. If cleared to '0', then the controller does not support the Timestamp feature. Refer to section 5.27.1.11. Bit 5 if set to '1', then the controller supports reservations. If cleared to '0', then the controller does not support reservations. If the controller supports reservations, then the following commands associated with reservations shall be supported: Reservation Report, Reservation Register, Reservation Acquire, and Reservation Release. Refer to section 8.19 for additional requirements. Bit 4 if set to '1', then the controller supports the Save field set to a non-zero value in the Set Features command and the Select field set to a non-zero value in the Get Features command. If cleared to '0', then the controller does not support the Save field set to a non-zero value in the Set Features command and the Select field set to a non-zero value in the Get Features command. Bit 3 if set to '1', then the controller supports the NVM Command Set Write Zeroes command and the Write Zeroes Size Limit (WZSL) field indicates the recommended maximum data size for Write Zeroes commands. If cleared to '0', then controller support of the NVM Command Set Write Zeroes command is indicated by a non-zero data size limit in the WZSL field. Bit 2 if set to '1', then the controller supports the NVM Command Set Dataset Management command and limits, if any, on controller support of the Dataset Management command are indicated by non-zero values in the Dataset Management Ranges Limit (DMRL) field, the Dataset Management Size Limit (DMSL) field and the Dataset Management Range Size Limit (DMRSL) field. If cleared to '0', then controller support of the NVM Command Set Dataset Management command is indicated by a non-zero data size limit in the DMRL, DMSL, and DMRSL fields. Bit 1 if set to '1', then the controller supports the NVM Command Set Write Uncorrectable command and the Write Uncorrectable Size Limit (WUSL) field indicates the recommended maximum data size for Write Uncorrectable commands. If cleared to '0', then controller support of the NVM Command Set Write Uncorrectable command is indicated by a non-zero data size limit in the WUSL field. Bit 0 if set to '1', then the controller supports the NVM Command Set Compare command. If cleared to '0', then the controller does not support the NVM Command Set Compare command. NOTE: This field applies to all I/O Command Sets. The original name has been retained for historical continuity.

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

Bytes	I/O[1]	Admin[1]	Disc[1]	Description
523:522	M	M	R	**Fused Operation Support (FUSES):** This field indicates the fused operations that the controller supports. Refer to section 3.4.2. Bits 15:1 are reserved. Bit 0 if set to '1', then the controller supports the NVM Command Set Compare and Write fused operation. If cleared to '0', then the controller does not support the NVM Command Set Compare and Write fused operation. Compare shall be the first command in the sequence.
524	M	M	R	**Format NVM Attributes (FNA):** This field indicates attributes for the Format NVM command. Bits 7:4 are reserved. Bit 3 indicates whether the Format NVM command supports an NSID value set to FFFFFFFFh. If set to '1', then the Format NVM command does not support an NSID value set to FFFFFFFFh. If cleared to '0', then the Format NVM command supports an NSID value set to FFFFFFFFh. Bit 2 indicates whether cryptographic erase is supported as part of the secure erase functionality. If set to '1', then cryptographic erase is supported. If cleared to '0', then cryptographic erase is not supported. Bit 1 indicates whether secure erase functionality applies to all namespaces in the NVM subsystem or is specific to a particular namespace. If set to '1', then any secure erase performed as part of a format operation results in a secure erase of all namespaces in the NVM subsystem. If cleared to '0', then any secure erase performed as part of a format results in a secure erase of the particular namespace specified. If bit 3 is set to '1', then this bit shall be cleared to '0'. Bit 0 indicates whether the format operation (excluding secure erase) applies to all namespaces in the NVM subsystem or is specific to a particular namespace. If set to '1', then all namespaces in the NVM subsystem shall be configured with the same attributes and a format (excluding secure erase) of any namespace results in a format of all namespaces in the NVM subsystem. If cleared to '0', then the controller supports format on a per namespace basis. If bit 3 is set to '1', then this bit shall be cleared to '0'.

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

<table>
<tr><th>Bytes</th><th>I/O[1]</th><th>Admin[1]</th><th>Disc[1]</th><th>Description</th></tr>
<tr><td>525</td><td>M</td><td>M</td><td>R</td><td>Volatile Write Cache (VWC): This field indicates attributes related to the presence of a volatile write cache in the controller.

Bits 7:3 are reserved.

Bits 2:1 indicate Flush command behavior (refer to the Flush Command section of the NVM Command Set Specification~~7.1~~) if the NSID value is set to FFFFFFFFh as follows:
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>Support for the NSID field set to FFFFFFFFh is not indicated. Only controllers compliant with NVM Express Base Specification revision 1.3 and earlier shall be allowed to return this value.</td></tr>
<tr><td>01b</td><td>Reserved.</td></tr>
<tr><td>10b</td><td>The Flush command does not support the NSID field set to FFFFFFFFh. The controller shall fail a Flush command with the NSID set to FFFFFFFFh with a status code of Invalid Namespace or Format.</td></tr>
<tr><td>11b</td><td>The Flush command supports the NSID field set to FFFFFFFFh.</td></tr>
</table>
Bit 0 if set to ‘1’ indicates that a volatile write cache is present. If cleared to ‘0’, a volatile write cache is not present.

If a volatile write cache is present, then the host controls whether the volatile write cache is enabled with a Set Features command specifying the Volatile Write Cache feature identifier (refer to section 5.27.1.4). The Flush command (refer to the Flush Command section of the NVM Command Set Specification~~7.1~~) is used to request that the contents of a volatile write cache be made non-volatile.</td></tr>
<tr><td>527:526</td><td>M</td><td>R</td><td>R</td><td>Atomic Write Unit Normal (AWUN): This field is specific to namespaces that are associated with command sets that specify logical blocks (i.e., Command Set Identifier 0h or 2h), and shall be cleared to 0h for namespaces that are not associated with command sets that specify logical blocks.</td></tr>
<tr><td>529:528</td><td>M</td><td>M</td><td>R</td><td>Atomic Write Unit Power Fail (AWUPF): This field is specific to namespaces that are associated with command sets that specify logical blocks (i.e., Command Set Identifier 0h or 2h), and shall be cleared to 0h for namespaces that are not associated with command sets that specify logical blocks.</td></tr>
<tr><td>530</td><td>M</td><td>M</td><td>R</td><td>I/O Command Set Vendor Specific Command Configuration (ICSVSCC): This field indicates the configuration settings for I/O Command Set Vendor Specific command handling. Refer to section 8.23.

Bits 7:1 are reserved.

Bit 0 if set to ‘1’ indicates that all NVM Vendor Specific Commands use the format defined in Figure 88. If cleared to ‘0’ indicates that the format of all NVM Vendor Specific Commands are vendor specific.</td></tr>
</table>

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

<table>
<tr><th>Bytes</th><th>I/O[1]</th><th>Admin[1]</th><th>Disc[1]</th><th>Description</th></tr>
<tr><td>531</td><td>M</td><td>M</td><td>R</td><td>Namespace Write Protection Capabilities (NWPC): This field indicates the optional namespace write protection capabilities supported by the controller. Refer to section 8.12.

Bits 7:3 are reserved.

Bit 2 if set to ‘1’, then the controller supports the Permanent Write Protect state. If cleared to ‘0’, then the controller does not support the Permanent Write Protect state. If this bit is set to ‘1’, then the controller shall support the Namespace Write Protection Authentication field (refer to section 8.18).

Bit 1 if set to ‘1’, then the controller supports the Write Protect Until Power Cycle state. If cleared to ‘0’, then the controller does not support Write Protect Until Power Cycle state. If this bit is set to ‘1’, then the controller shall support the Namespace Write Protection Authentication field (refer to section 8.18).

Bit 0 if set to ‘1’, then the controller shall support the No Write Protect and Write Protect namespace write protection states and may support the Write Protect Until Power Cycle state and Permanent Write Protect namespace write protection states (refer to section 8.12). If cleared to ‘0’, then the controller does not support Namespace Write Protection and bits 2:1 shall be cleared to 00b.</td></tr>
<tr><td>533:532</td><td>O</td><td>R</td><td>R</td><td>Atomic Compare & Write Unit (ACWU): This field is specific to namespaces that are associated with command sets that specify logical blocks (i.e., Command Set Identifier 0h or 2h), and shall be cleared to 0h for namespaces that are not associated with command sets that specify logical blocks.</td></tr>
<tr><td>535:534</td><td>M</td><td>R</td><td>R</td><td>Copy Descriptor Formats Supported:

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>15:2</td><td>Reserved</td></tr>
<tr><td>1</td><td>If set to ‘1’, then the controller supports Copy Descriptor Format 1h. If cleared to ‘0’, then the controller does not support Copy Descriptor Format 1h.</td></tr>
<tr><td>0</td><td>If set to ‘1’, then the controller supports Copy Descriptor Format 0h. If cleared to ‘0’, then the controller does not support Copy Descriptor Format 0h.</td></tr>
</table>
</td></tr>
</table>

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

<table>
<tr><th>Bytes</th><th>I/O[1]</th><th>Admin[1]</th><th>Disc[1]</th><th>Description</th></tr>
<tr><td>539:536</td><td>O</td><td>O</td><td>M</td><td>SGL Support (SGLS): This field indicates if SGLs are supported and the particular SGL types supported. Refer to section 4.1.2.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:22</td><td>Reserved</td></tr>
<tr><td>21</td><td>If set to ‘1’, then the controller supports the Transport SGL Data Block descriptor. If cleared to ‘0’, then the controller does not support the Transport SGL Data Block descriptor.</td></tr>
<tr><td>20</td><td>If set to ‘1’, then the controller supports the Address field in SGL Data Block, SGL Segment, and SGL Last Segment descriptor types specifying an offset. If cleared to ‘0’, then the Address field specifying an offset is not supported.</td></tr>
<tr><td>19</td><td>If set to ‘1’, then use of a Metadata Pointer (MPTR) that contains an address of an SGL segment containing exactly one SGL Descriptor that is qword aligned is supported. If cleared to ‘0’, then use of a MPTR containing an SGL Descriptor is not supported.</td></tr>
<tr><td>18</td><td>If set to ‘1’, then the controller supports commands that contain a data or metadata SGL of a length larger than the amount of data to be transferred. If cleared to ‘0’, then the SGL length shall be equal to the amount of data to be transferred.</td></tr>
<tr><td>17</td><td>This field specifies metadata buffer alignment requirements when CDW0.PSDT is set to 01b (refer to Figure 86). If set to ‘1’, then use of a byte aligned contiguous physical buffer of metadata (the Metadata Pointer field in Figure 87) is supported. If cleared to ‘0’, then use of a byte aligned contiguous physical buffer of metadata is not supported.</td></tr>
<tr><td>16</td><td>If set to ‘1’, then the SGL Bit Bucket descriptor is supported. If cleared to ‘0’, then the SGL Bit Bucket descriptor is not supported.</td></tr>
<tr><td>15:08</td><td>SGL Descriptor Threshold (SDT): This field indicates the recommended maximum number of SGL descriptors in a command (refer to section 4.1.2). If this field is cleared to 0h, then no recommended maximum number of SGL descriptors is reported.</td></tr>
<tr><td>07:03</td><td>Reserved</td></tr>
<tr><td>02</td><td>If set to ‘1’, then the controller supports the Keyed SGL Data Block descriptor. If cleared to ‘0’, then the controller does not support the Keyed SGL Data Block descriptor.</td></tr>
<tr><td>01:00</td><td>This field is used to determine the SGL support. Valid values are shown in the table below.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>SGLs are not supported.</td></tr>
<tr><td>01b</td><td>SGLs are supported. There is no alignment nor granularity requirement for Data Blocks.</td></tr>
<tr><td>10b</td><td>SGLs are supported. There is a dword alignment and granularity requirement for Data Blocks (refer to section 4.1.2).</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table></td></tr>
</table></td></tr>
<tr><td>543:540</td><td>O</td><td>O</td><td>R</td><td>Maximum Number of Allowed Namespaces (MNAN): This field indicates the maximum number of namespaces supported by the NVM subsystem. If this field is cleared to 0h, then the maximum number of namespaces supported by the NVM subsystem is less than or equal to the value in the NN field. If the controller supports Asymmetric Namespace Access Reporting, then this field shall be set to a non-zero value that is less than or equal to the NN value.</td></tr>
</table>

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

Bytes	I/O[1]	Admin[1]	Disc[1]	Description
559:544	O	R	R	**Maximum Domain Namespace Attachments (MAXDNA):** Indicates the maximum of the sum of the number of namespaces attached to each I/O controller in the Domain. If this field is cleared to 0h, then no maximum is specified. The value of this field shall be the same value for all I/O controllers in the Domain.
563:560	O	R	R	**Maximum I/O Controller Namespace Attachments (MAXCNA):** Indicates the maximum number of namespaces that are allowed to be attached to this I/O controller. If this field is cleared to 0h, then no maximum is specified. The value of this field shall be less than or equal to the number of namespaces supported by the NVM subsystem (refer to the MNAN field).
767:564				Reserved
1023:768	M	M	R	**NVM Subsystem NVMe Qualified Name (SUBNQN):** This field specifies the NVM Subsystem NVMe Qualified Name as a UTF-8 null-terminated string. Refer to section 4.5 for the definition of NVMe Qualified Name. Support for this field is mandatory if the controller supports revision 1.2.1 or later as indicated in the Version property (refer to section 3.1.3.2).
1791:1024				Reserved
Fabric Specific				
1795:1792	M[2]	M[2]	R	**I/O Queue Command Capsule Supported Size (IOCCSZ):** This field defines the maximum I/O command capsule size in 16 byte units. The minimum value that shall be indicated is 4 corresponding to 64 bytes.
1799:1796	M[2]	M[2]	R	**I/O Queue Response Capsule Supported Size (IORCSZ):** This field defines the maximum I/O response capsule size in 16 byte units. The minimum value that shall be indicated is 1 corresponding to 16 bytes.
1801:1800	M[2]	M[2]	R	**In Capsule Data Offset (ICDOFF):** This field defines the offset where data starts within a capsule. This value is applicable to I/O Queues only (the Admin Queue shall use a value of 0h). The value is specified in 16 byte units. The offset is from the end of the submission queue entry within the command capsule (starting at 64 bytes in the command capsule). The minimum value is 0 and the maximum value is FFFFh.
1802	M[2]	M[2]	R	**Fabrics Controller Attributes (FCATT):** This field indicates attributes of the controller that are specific to NVMe over Fabrics. Bits 7:1 are reserved. Bit 0 if cleared to '0', then the NVM subsystem uses a dynamic controller model. Bit 0 if set to '1', then the NVM subsystem uses a static controller model.
1803	M[2]	M[2]	R	**Maximum SGL Data Block Descriptors (MSDBD):** This field indicates the maximum number of SGL Data Block or Keyed SGL Data Block descriptors that a host is allowed to place in a capsule. A value of 0h indicates no limit.
1805:1804	M[2]	M[2]	R	**Optional Fabric Commands Support (OFCS):** Indicate whether the controller supports optional fabric commands. Bits 15:1 are reserved. Bit 0 if cleared to '0' then the controller does not support the Disconnect command. Bit 0 if set to '1' then the controller supports the Disconnect command and deletion of individual I/O Queues.
2047:1806				Reserved
Power State Descriptors				
2079:2048	M	M	R	**Power State 0 Descriptor (PSD0):** This field indicates the characteristics of power state 0. The format of this field is defined in Figure 276.
2111:2080	O	O	R	**Power State 1 Descriptor (PSD1):** This field indicates the characteristics of power state 1. The format of this field is defined in Figure 276.
2143:2112	O	O	R	**Power State 2 Descriptor (PSD2):** This field indicates the characteristics of power state 2. The format of this field is defined in Figure 276.

Figure 275: Identify – Identify Controller Data Structure, I/O Command Set Independent

Bytes	I/O[1]	Admin[1]	Disc[1]	Description
…				
3071:3040	O	O	R	**Power State 31 Descriptor (PSD31):** This field indicates the characteristics of power state 31. The format of this field is defined in Figure 276.
Vendor Specific				
4095:3072	O	O	O	Vendor Specific.

Notes:
1. O/M/R definition: O = Optional, M = Mandatory, R = Reserved.
2. Mandatory for I/O controllers using a message-based transport. Reserved for controllers using a memory-based transport.

Figure 276 defines the power state descriptor that describes the attributes of each power state. For more information on how the power state descriptor fields are used, refer to section 8.15 on power management.

Figure 276: Identify – Power State Descriptor Data Structure

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>255:184</td><td>Reserved</td></tr>
<tr><td>183:182</td><td>Active Power Scale (APS): This field indicates the scale for the Active Power field. If an Active Power Workload is reported for a power state, then the Active Power Scale shall also be reported for that power state.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>Not reported for this power state</td></tr>
<tr><td>01b</td><td>0.0001 W</td></tr>
<tr><td>10b</td><td>0.01 W</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>181:179</td><td>Reserved</td></tr>
<tr><td>178:176</td><td>Active Power Workload (APW): This field indicates the workload used to calculate maximum power for this power state. Refer to section 8.15.3 for more details on each of the defined workloads. This field shall not be “No Workload” unless ACTP is 0h.</td></tr>
<tr><td>175:160</td><td>Active Power (ACTP): This field indicates the largest average power consumed by the NVM subsystem over a 10 second period in this power state with the workload indicated in the Active Power Workload field. The power in Watts is equal to the value in this field multiplied by the scale indicated in the Active Power Scale field. A value of 0h indicates Active Power is not reported.</td></tr>
<tr><td>159:152</td><td>Reserved</td></tr>
<tr><td>151:150</td><td>Idle Power Scale (IPS): This field indicates the scale for the Idle Power field.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>Not reported for this power state</td></tr>
<tr><td>01b</td><td>0.0001 W</td></tr>
<tr><td>10b</td><td>0.01 W</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>149:144</td><td>Reserved</td></tr>
<tr><td>143:128</td><td>Idle Power (IDLP): This field indicates the typical power consumed by the NVM subsystem over 30 seconds in this power state when idle (e.g., there are no pending commands, property accesses, background processes, sanitize operation, nor device self-test operations). The measurement starts after the NVM subsystem has been idle for 10 seconds. The power in Watts is equal to the value in this field multiplied by the scale indicated in the Idle Power Scale field. A value of 0h indicates Idle Power is not reported. Refer to section 8.15.
Note: This value may be used by hosts to manage power versus resume latency. Platform and form factor specifications may have additional power measurement and reporting requirements that are outside the scope of this specification.</td></tr>
<tr><td>127:125</td><td>Reserved</td></tr>
</table>

Figure 276: Identify – Power State Descriptor Data Structure

Bits	Description
124:120	**Relative Write Latency (RWL):** This field indicates the relative write latency associated with this power state. The value in this field shall be less than the number of supported power states (e.g., if the controller supports 16 power states, then valid values are 0 through 15). A lower value means lower write latency.
119:117	Reserved
116:112	**Relative Write Throughput (RWT):** This field indicates the relative write throughput associated with this power state. The value in this field shall be less than the number of supported power states (e.g., if the controller supports 16 power states, then valid values are 0 through 15). A lower value means higher write throughput.
111:109	Reserved
108:104	**Relative Read Latency (RRL):** This field indicates the relative read latency associated with this power state. The value in this field shall be less than the number of supported power states (e.g., if the controller supports 16 power states, then valid values are 0 through 15). A lower value means lower read latency.
103:101	Reserved
100:96	**Relative Read Throughput (RRT):** This field indicates the relative read throughput associated with this power state. The value in this field shall be less than the number of supported power states (e.g., if the controller supports 16 power states, then valid values are 0 through 15). A lower value means higher read throughput.
95:64	**Exit Latency (EXLAT):** This field indicates the maximum exit latency in microseconds associated with exiting this power state. A value of 0h indicates Exit Latency is not reported.
63:32	**Entry Latency (ENLAT):** This field indicates the maximum entry latency in microseconds associated with entering this power state. A value of 0h indicates Entry Latency is not reported.
31:26	Reserved
25	**Non-Operational State (NOPS):** This bit indicates whether the controller processes I/O commands in this power state. If this bit is cleared to '0', then the controller processes I/O commands in this power state. If this bit is set to '1', then the controller does not process I/O commands in this power state. Refer to section 8.15.1.
24	**Max Power Scale (MXPS):** This bit indicates the scale for the Maximum Power field. If this bit is cleared to '0', then the scale of the Maximum Power field is in 0.01 Watts. If this bit is set to '1', then the scale of the Maximum Power field is in 0.0001 Watts.
23:16	Reserved
15:00	**Maximum Power (MP):** This field indicates the sustained maximum power consumed by the NVM subsystem in this power state. The power in Watts is equal to the value in this field multiplied by the scale specified in the Max Power Scale bit. A value of 0h indicates Maximum Power is not reported. Refer to section 8.15. Note: This value is intended to provide an approximate guideline for hosts to manage power versus performance. Platform and form factor specifications may have additional power measurement and reporting requirements that are outside the scope of this specification.

5.17.2.2 Active Namespace ID list (CNS 02h)

A list of 1,024 namespace IDs is returned to the host containing active NSIDs in increasing order that are greater than the value specified in the Namespace Identifier (NSID) field of the command. The controller should abort the command with a status code of Invalid Namespace or Format if the NSID field is set to FFFFFFFEh or FFFFFFFFh. The NSID field may be cleared to 0h to retrieve a Namespace List including the namespace starting with NSID of 1h. The data structure returned is a Namespace List (refer to section 4.4.2).

5.17.2.3 Namespace Identification Descriptor list (CNS 03h)

A list of Namespace Identification Descriptor structures (refer to Figure 277) is returned to the host for the namespace specified in the Namespace Identifier (NSID) field if it is an active NSID. Namespace Identification Descriptor structures consist of one or more Namespace Identifiers (NID) of various types as indicated by the Namespace Identifier Type (NIDT) field in each descriptor. Each NID is assigned to a

namespace at namespace creation and remains fixed throughout the life of that namespace. If the NSID field does not specify an active NSID, then refer to section 3.2.1.5 for the status code to return.

The contents of the Namespace Identification Descriptor list is preserved across namespace and controller operations (e.g., Controller Level Reset, namespace format, etc.).

The controller may return any number of variable length Namespace Identification Descriptor structures that fit into the 4,096 byte Identify payload. All remaining bytes after the Namespace Identification Descriptor structures should be cleared to 0h, and the host shall interpret a Namespace Identifier Descriptor Length (NIDL) value of 0h as the end of the list. The host should ignore any Namespace Identification Descriptor with a Namespace Identifier Type not supported by the host.

A controller shall not return multiple Namespace Identification Descriptors with the same Namespace Identifier Type (NIDT). A controller shall return:

- at least one Namespace Identification Descriptor identifying the namespace (i.e., NIDT field set to 1h, 2h, or 3h); and
- a Namespace Identifcation Descriptor identifying the I/O Command Set (i.e., NIDT field set to 4h) if CAP.CSS bit 6 is set to '1'.

Figure 277: Identify – Namespace Identification Descriptor

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>00</td><td>Namespace Identifier Type (NIDT): This field indicates the data type contained in the Namespace Identifier field and the length of that type as defined in the following table.
<table>
<tr><th>Value</th><th>Length (NIDL)</th><th>Definition</th></tr>
<tr><td>0h</td><td></td><td>Reserved</td></tr>
<tr><td>1h</td><td>8h</td><td>IEEE Extended Unique Identifier: The NID field contains a copy of the EUI64 field in the Identify Namespace data structure (refer to the applicable I/O Command Set specification). If the EUI64 field of the Identify Namespace data structure is not supported, (i.e., EUI64 field is cleared to 0h), the controller shall not report a Namespace Identification Descriptor with a value of type 1h.</td></tr>
<tr><td>2h</td><td>10h</td><td>Namespace Globally Unique Identifier: The NID field contains a copy of the NGUID field in the Identify Namespace data structure (refer to the applicable I/O Command Set specification). If the NGUID field of the Identify Namespace data structure is not supported (i.e., the NGUID field is cleared to 0h), the controller shall not report a Namespace Identification Descriptor with a value of type 2h.</td></tr>
<tr><td>3h</td><td>10h</td><td>Namespace UUID: The NID field contains a 128-bit Universally Unique Identifier (UUID) as specified in RFC 4122. Refer to section 4.3.6.
If the namespace does not support an IEEE Extended Unique Identifier (i.e., EUI64 field is cleared to 0h) and does not support a Namespace Globally Unique Identifier (i.e., the NGUID field is cleared to 0h), then the namespace shall report a Namespace Identification Descriptor with a value of type 3h.</td></tr>
<tr><td>4h</td><td>1h</td><td>Command Set Identifier (CSI): The NID field contains the I/O Command Set that operates on this namespace. Refer to Figure 274.</td></tr>
<tr><td>5h to FFh</td><td></td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>01</td><td>Namespace Identifier Length (NIDL): This field contains the length in bytes of the Namespace Identifier (NID) field. The total length of the Namespace Identification Descriptor in bytes is the value in this field plus four. If this field is cleared to 0h it indicates the end of the Namespace Identifier Descriptor list.</td></tr>
<tr><td>02:03</td><td>Reserved</td></tr>
<tr><td>(NIDL + 3):04</td><td>Namespace Identifier (NID): For an NIDT field value of 1h, 2h, and 3h, this field contains a value that is globally unique. The type of the value is specified by the Namespace Identifier Type (NIDT) field, and the size is specified by the Namespace Identifier Length (NIDL) field.</td></tr>
</table>

5.17.2.4 NVM Set List (CNS 04h)

Figure 278 defines an NVM Set List. The data structure is an ordered list of NVM Set Attribute Entry data structures, sorted by NVM Set Identifier, starting with the first NVM Set Identifier supported by the NVM subsystem that is equal to or greater than the NVM Set Identifier indicated in CDW11.NVMSETID and are accessible by the controller processing the command. The NVM Set List describes the attributes for each NVM Set in the list based on the NVM Set Attributes Entry in Figure 278.

The NVM Set List shall not contain an entry cleared to 0h.

Figure 278: NVM Set List

Bytes	Description
00	**Number of Identifiers:** This field indicates the number of NVM Set Attributes Entries in the list. There are up to 31 entries in the list. A value of 0h indicates that there are no entries in the list.
127:01	Reserved
255:128	**Entry 0:** This field contains the first NVM Set Attributes Entry in the list, if present.
383:256	**Entry 1:** This field contains the second NVM Set Attributes Entry in the list, if present.
…	**…**
(N*128+255): (N*128+128)	**Entry N:** This field contains the N+1 NVM Set Attributes Entry in the list, if present.

Figure 279: NVM Set Attributes Entry

Bytes	Description
01:00	**NVM Set Identifier:** This field indicates the identifier of the NVM Set in the NVM subsystem that is described by this entry.
03:02	**Endurance Group Identifier:** This field indicates the Endurance Group for this NVM Set. Refer to section 3.2.3.
07:04	Reserved
11:08	**Random 4 KiB Read Typical:** This field indicates the typical time to complete a 4 KiB random read in 100 nanosecond units when the NVM Set is in a Predictable Latency Mode Deterministic Window and there is 1 outstanding command per NVM Set.
15:12	**Optimal Write Size:** This field indicates the size in bytes for optimal write performance. A value of 0h indicates that no Optimal Write Size is specified. This field should be cleared to 0h when namespaces within an NVM Set have different User Data Formats that do not allow an Optimal Write Size to be specified.
31:16	**Total NVM Set Capacity:** This field indicates the total NVM capacity in this NVM Set. The value is in bytes.
47:32	**Unallocated NVM Set Capacity:** This field indicates the unallocated NVM capacity in this NVM Set. The value is in bytes.
127:48	Reserved

5.17.2.5 I/O Command Set specific Identify Namespace data structure (CNS 05h)

An I/O Command Set specific Identify Namespace data structure (refer to the applicable I/O Command Set specification) is returned to the host for the namespace specified in the Namespace Identifier (NSID) field if the NSID is active. If the specified namespace is an inactive NSID, then the controller returns a zero filled data structure.

The specific Identify Namespace data structure that is returned by this command is specified by the Command Set Identifier (CSI) field (refer to Figure 274). If the I/O Command Set associated with the namespace identified by the NSID field does not support the Identify Namespace data structure specified by the CSI field, the controller shall abort the command with a status code of Invalid Field in Command.

If the controller supports the Namespace Management capability (refer to section 8.11), the I/O Command Set requested in the CSI field is enabled (refer to CC.CSS in Figure 46), and the NSID field is set to FFFFFFFFh, then the controller returns an I/O Command Set specific Identify Namespaces data structure

that specifies capabilities that are common across namespaces for the I/O Command Set specified in the CSI field (refer to Figure 274).

If the controller does not support the Namespace Management capability and the NSID field is set to FFFFFFFFh, then the controller shall abort the command with a status code of Invalid Namespace or Format.

5.17.2.6 I/O Command Set specific Identify Controller data structure (CNS 06h)

An I/O Command Set specific Identify Controller data structure is returned to the host for the controller processing the command. The specific Identify Controller data structure that is returned by this command is specified by the Command Set Identifier (CSI) field (refer to Figure 274). Data structures for specific I/O Command Sets are optionally defined by the I/O Command Set specifications. If the I/O Command Set specified by the CSI field does not have an Identify Controller data structure, then the controller shall return a zero filled data structure. If the host requests a data structure for an I/O Command Set that the controller does not support, the controller shall abort the command with a status code of Invalid Field in Command.

5.17.2.7 I/O Command Set specific Active Namespace ID list (CNS 07h)

A list of 1,024 namespace IDs is returned to the host containing active NSIDs in increasing order that are greater than the value specified in the Namespace Identifier (NSID) field of the command as specified by the Command Set Identifier (CSI) field of the command. Only namespaces associated with the I/O Command Set specified by the CSI value are returned. For CSI values that are not supported or not enabled the command is aborted with a status code of Invalid Field in Command.

The controller should abort the command with a status code of Invalid Namespace or Format if the NSID field is set to FFFFFFFEh or FFFFFFFFh. The NSID field may be cleared to 0h to retrieve a Namespace List including the namespace starting with NSID of 1h. The data structure returned is a Namespace List (refer to section 4.4.2).

5.17.2.8 I/O Command Set Independent Identify Namespace data structure (CNS 08h)

If the Namespace Identifier (NSID) field specifies an active NSID, then the I/O Command Set Independent Identify Namespace data structure (refer to Figure 280) is returned to the host for that specified namespace. If that specified namespace is an inactive NSID, then the controller returns a zero filled data structure.

If the controller supports the Namespace Management capability (refer to section 8.11) and the NSID field is set to FFFFFFFFh, then the controller returns an I/O Command Set Independent Identify Namespace data structure that specifies capabilities that are common for the controller. If the controller does not support the Namespace Management capability and the NSID field is set to FFFFFFFFh, then the controller shall abort the command with a status code of Invalid Namespace or Format.

Figure 280: Identify – I/O Command Set Independent Identify Namespace Data Structure

Bytes	O/M [1]	Description
00	M	**Common Namespace Features (NSFEAT):** This field defines features of the namespace. Bits 7:5 are reserved. Bit 4 **Rotational Media** (**RMEDIA**) if set to '1' indicates that the namespace stores data on rotational media (refer to section 8.20). If cleared to '0', indicates that the namespace does not store data on rotational media. Bit 3 (**UIDREUSE**) if set to '1' indicates that the value in the NGUID field for this namespace, if non-zero, is never reused by the controller and that the value in the EUI64 field for this namespace, if non-zero, is never reused by the controller. If cleared to '0', then the NGUID value may be reused and the EUI64 value may be reused by the controller for a new namespace created after this namespace is deleted. This bit shall be cleared to '0' if both NGUID and EUI64 fields are cleared to 0h. Refer to section 4.5.1. Bit 2:0 are reserved.
01	O	**Namespace Multi-path I/O and Namespace Sharing Capabilities (NMIC):** This field specifies multi-path I/O and namespace sharing capabilities of the namespace. Bits 7:1 are reserved. Bit 0: If set to '1', then the namespace may be attached to two or more controllers in the NVM subsystem concurrently (i.e., may be a shared namespace). If cleared to '0', then the namespace is a private namespace and is able to be attached to only one controller at a time.

Figure 280: Identify – I/O Command Set Independent Identify Namespace Data Structure

Bytes	O/M [1]	Description
02	O	**Reservation Capabilities (RESCAP):** This field indicates the reservation capabilities of the namespace. A value of 0h in this field indicates that reservations are not supported by this namespace. Refer to section 8.19 for more details. Bit 7 if set to '1' indicates that Ignore Existing Key is used as defined in NVM Express Base Specification revision 1.3 or later. Bit 7 if cleared to '0' indicates that Ignore Existing Key is used as defined in NVM Express Base Specification revision 1.2.1 or earlier. This bit shall be set to '1' if the controller supports revision 1.3 or later as indicated in the Version register. Bit 6 if set to '1' indicates that the namespace supports the Exclusive Access – All Registrants reservation type. If this bit is cleared to '0', then the namespace does not support the Exclusive Access – All Registrants reservation type. Bit 5 if set to '1' indicates that the namespace supports the Write Exclusive – All Registrants reservation type. If this bit is cleared to '0', then the namespace does not support the Write Exclusive – All Registrants reservation type. Bit 4 if set to '1' indicates that the namespace supports the Exclusive Access – Registrants Only reservation type. If this bit is cleared to '0', then the namespace does not support the Exclusive Access – Registrants Only reservation type. Bit 3 if set to '1' indicates that the namespace supports the Write Exclusive – Registrants Only reservation type. If this bit is cleared to '0', then the namespace does not support the Write Exclusive – Registrants Only reservation type. Bit 2 if set to '1' indicates that the namespace supports the Exclusive Access reservation type. If this bit is cleared to '0', then the namespace does not support the Exclusive Access reservation type. Bit 1 if set to '1' indicates that the namespace supports the Write Exclusive reservation type. If this bit is cleared to '0', then the namespace does not support the Write Exclusive reservation type. Bit 0 if set to '1' indicates that the namespace supports the Persist Through Power Loss capability. If this bit is cleared to '0', then the namespace does not support the Persist Through Power Loss Capability.
03	O	**Format Progress Indicator (FPI):** If a format operation is in progress, this field indicates the percentage of the namespace that remains to be formatted. Bit 7 if set to '1' indicates that the namespace supports the Format Progress Indicator defined by bits 6:0 in this field. If this bit is cleared to '0', then the namespace does not support the Format Progress Indicator and bits 6:0 in this field shall be cleared to 0h. Bits 6:0 indicate the percentage of the Format NVM command that remains to be completed (e.g., a value of 25 indicates that 75% of the Format NVM command has been completed and 25% remains to be completed). If bit 7 is set to '1', then a value of 0h indicates that the namespace is formatted with the format specified by Identify Namespace data structures (refer to section 1.5.29) and there is no Format NVM command in progress.
07:04	O	**ANA Group Identifier (ANAGRPID):** For NSID other than FFFFFFFFh, this field indicates the ANA Group Identifier of the ANA group (refer to section 8.1.2) of which the namespace is a member. Each namespace that is attached to a controller that supports Asymmetric Namespace Access Reporting (refer to the CMIC field) shall report a valid ANAGRPID. If the controller does not support Asymmetric Namespace Access Reporting, then this field shall be cleared to 0h. If the value in this field changes and Asymmetric Namespace Access Change Notices are supported and enabled, then the controller shall issue an Asymmetric Namespace Access Change Notice.

Figure 280: Identify – I/O Command Set Independent Identify Namespace Data Structure

<table>
<tr><th>Bytes</th><th>O/M [1]</th><th>Description</th></tr>
<tr><td>08</td><td>O</td><td>Namespace Attributes (NSATTR): This field specifies attributes of the namespace.

Bits 7:1 are reserved.

Bit 0: If set to ‘1’, then the namespace is currently write protected due to any condition (e.g., namespace write protection set for the namespace, media errors) and all write access to the namespace shall fail. If cleared to ‘0’, then the namespace is not currently write protected.</td></tr>
<tr><td>09</td><td></td><td>Reserved</td></tr>
<tr><td>11:10</td><td>O</td><td>NVM Set Identifier (NVMSETID): For NSID other than FFFFFFFFh, this field indicates the NVM Set with which this namespace is associated. If NVM Sets are not supported by the controller, then this field shall be cleared to 0h.</td></tr>
<tr><td>13:12</td><td>O</td><td>Endurance Group Identifier (ENDGID): For NSID other than FFFFFFFFh, this field indicates the Endurance Group with which this namespace is associated. If Endurance Groups are not supported by the controller, then this field shall be cleared to 0h.</td></tr>
<tr><td>14</td><td>M</td><td>Namespace Status (NSTAT): This field indicates the status of the namespace with the specified NSID.

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:1</td><td>Reserved</td></tr>
<tr><td>0</td><td>Namespace Ready (NRDY): A value of ‘1’ indicates that the namespace is ready (refer to section 3.5.3). A value of ‘0’ indicates that the namespace is not ready.</td></tr>
</table>
</td></tr>
<tr><td>4095:15</td><td></td><td>Reserved</td></tr>
<tr><td colspan="3">Notes:
1. O/M definition: O = Optional, M = Mandatory.</td></tr>
</table>

5.17.2.9 Allocated Namespace ID list (CNS 10h)

A list of up to 1,024 namespace IDs is returned to the host containing allocated NSIDs in increasing order that are greater than the value specified in the Namespace Identifier (NSID) field of the Identify command.

The controller should abort the command with a status code of Invalid Namespace or Format if the NSID field is set to FFFFFFFEh or FFFFFFFFh. The NSID field may be cleared to 0h to retrieve a Namespace List including the namespace starting with NSID of 1h. The data structure returned is a Namespace List (refer to section 4.4.2).

5.17.2.10 Identify Namespace data structure for an Allocated Namespace ID (CNS 11h)

The Identify Namespace data structure (refer to the NVM Command Set Specification) is returned to the host for the namespace specified in the Namespace Identifier (NSID) field if it is an allocated NSID. If the specified namespace is an unallocated NSID, then the controller returns a zero filled data structure. If the specified namespace is not associated with an I/O Command Set that specifies logical blocks (e.g., the NVM Command Set), then the controller shall abort the command with a status code of Invalid I/O Command Set.

If the specified namespace is an invalid NSID, then the controller shall abort the command with a status code of Invalid Namespace or Format. If the NSID field is set to FFFFFFFFh, then the controller should abort the command with a status code of Invalid Namespace or Format.

5.17.2.11 Namespace Attached Controller list (CNS 12h)

A Controller List (refer to section 4.4.1) of up to 2,047 controller identifiers is returned containing a controller identifier greater than or equal to the value specified in the Controller Identifier (CDW10.CNTID) field. The list contains controller identifiers of controllers that are attached to the namespace specified in the

Namespace Identifier (NSID) field. If the NSID field is set to FFFFFFFFh, then the controller should abort the command with a status code of Invalid Field in Command.

5.17.2.12 Controller list (CNS 13h)

A Controller List (refer to section 4.4.1) of up to 2,047 controller identifiers is returned containing a controller identifier greater than or equal to the value specified in the Controller Identifier (CDW10.CNTID) field. The list contains controller identifiers of controllers in the NVM subsystem that are capable of being attached to namespace(s).

5.17.2.13 Primary Controller Capabilities data structure (CNS 14h)

The Primary Controller Capabilities Structure (refer to Figure 281) is returned to the host for the primary controller specified.

Figure 281: Identify – Primary Controller Capabilities Structure

Bytes	Description
01:00	**Controller Identifier (CNTLID):** This field indicates the Controller Identifier of the primary controller.
03:02	**Port Identifier (PORTID):** This field indicates the Port Identifier of the NVM subsystem port associated with the primary controller. The Port Identifier for a PCI Express Port shall be unique within the NVM subsystem. If the NVM subsystem supports an NVMe-MI Management Endpoint on this PCIe port, then this field shall contain the same value as the Port Identifier field in the Controller Information Data Structure (refer to the NVM Express Management Interface Specification) for this primary controller.
04	**Controller Resource Types (CRT):** This field indicates the controller resources types supported. If a primary controller supports a controller resource type, then all associated secondary controllers shall support that controller resource type. Bits 7:2 are reserved. Bit 1 if set to '1', then VI Resources are supported. Bit 1 if cleared to '0', then VI Resources are not supported. Refer to section 8.26.2. Bit 0 if set to '1', then VQ Resources are supported. Bit 0 if cleared to '0', then VQ Resources are not supported. Refer to section 8.26.1.
31:05	Reserved
35:32	**VQ Resources Flexible Total (VQFRT):** This field indicates the total number of VQ Flexible Resources for the primary and its secondary controllers.
39:36	**VQ Resources Flexible Assigned (VQRFA):** This field indicates the total number of VQ Flexible Resources Assigned to the associated secondary controllers.
41:40	**VQ Resources Flexible Allocated to Primary (VQRFAP):** This field indicates the total number of VQ Flexible Resources currently allocated to the primary controller. This value may change after a Controller Level Reset other than a Controller Reset (i.e., CC.EN transitions from '1' to '0') if a new value was set using the Virtualization Management command. The default value of this field is implementation specific.
43:42	**VQ Resources Private Total (VQPRT):** This field indicates the total number of VQ Private Resources for the primary controller.
45:44	**VQ Resources Flexible Secondary Maximum (VQFRSM):** This field indicates the maximum number of VQ Flexible Resources that may be assigned to a secondary controller.
47:46	**VQ Flexible Resource Preferred Granularity (VQGRAN):** This field indicates the preferred granularity of assigning and removing VQ Flexible Resources. Assigning and removing VQ Resources in this granularity minimizes any wasted internal implementation resources.
63:48	Reserved
67:64	**VI Resources Flexible Total (VIFRT):** This field indicates the total number of VI Flexible Resources for the primary and its secondary controllers.
71:68	**VI Resources Flexible Assigned (VIRFA):** This field indicates the total number of VI Flexible Resources Assigned to the associated secondary controllers.

Figure 281: Identify – Primary Controller Capabilities Structure

Bytes	Description
73:72	**VI Resources Flexible Allocated to Primary (VIRFAP):** This field indicates the total number of VI Flexible Resources currently allocated to the primary controller. This value may change after a Controller Level Reset other than a Controller Reset (i.e., CC.EN transitions from '1' to '0') if a new value was set using the Virtualization Management command. The default value of this field is implementation specific.
75:74	**VI Resources Private Total (VIPRT):** This field indicates the total number of VI Private Resources for the primary controller.
77:76	**VI Resources Flexible Secondary Maximum (VIFRSM):** This field indicates the maximum number of VI Flexible Resources that may be assigned to a secondary controller.
79:78	**VI Flexible Resource Preferred Granularity (VIGRAN):** This field indicates the preferred granularity of assigning and removing VI Flexible Resources. Assigning and removing VI Resources in this granularity minimizes any wasted internal implementation resources.
4095:80	Reserved

5.17.2.14 Secondary Controller list (CNS 15h)

A Secondary Controller List (refer to Figure 282) is returned to the host for up to 127 secondary controllers associated with the primary controller processing this command. The list contains entries for controller identifiers greater than or equal to the value specified in the Controller Identifier (CDW10.CNTID) field.

All secondary controllers are represented, including those that are in an Offline state due to SR-IOV configuration settings (e.g., VF Enable is cleared to '0' or NumVFs specifies a value that does not enable the associated secondary controller).

Figure 282: Secondary Controller List

Bytes	Description
00	**Number of Identifiers:** This field indicates the number of Secondary Controller Entries in the list. There are up to 127 entries in the list. A value of 0h indicates there are no entries in the list.
31:01	Reserved
63:32	**SC Entry 0:** This field contains the first Secondary Controller Entry in the list, if present.
95:64	**SC Entry 1:** This field contains the second Secondary Controller Entry in the list, if present.
…	…
(N*32+63): (N*32+32)	**SC Entry N:** This field contains the N+1 Secondary Controller Entry in the list, if present.

Figure 283: Secondary Controller Entry

Bytes	Description
01:00	**Secondary Controller Identifier (SCID):** This field indicates the Controller Identifier of the secondary controller described by this entry.
03:02	**Primary Controller Identifier (PCID):** This field indicates the Controller Identifier of the associated primary controller.
04	**Secondary Controller State (SCS):** This field indicates the state of the secondary controller. Bits 7:1 are reserved. Bit 0 if set to '1', then the controller is in the Online state. Bit 0 if cleared to '0', then the controller is in the Offline state.
07:05	Reserved
09:08	**Virtual Function Number (VFN):** If the secondary controller is an SR-IOV VF, this field indicates its VF Number, where VF Number > 0, and VF Number is no larger than the total number of VFs indicated by the TotalVFs register (refer to Single Root I/O Virtualization and Sharing Specification) in the PF's SR-IOV Extended Capability structure. If the secondary controller is not an SR-IOV VF, then this field is cleared to 0h.
11:10	**Number of VQ Flexible Resources Assigned (NVQ):** This field indicates the number of VQ Flexible Resources currently assigned to the indicated secondary controller.

Figure 283: Secondary Controller Entry

Bytes	Description
13:12	**Number of VI Flexible Resources Assigned (NVI):** This field indicates the number of VI Flexible Resources currently assigned to the indicated secondary controller.
31:14	Reserved

5.17.2.15 Namespace Granularity List (CNS 16h)

If the controller supports reporting of Namespace Granularity refer to the applicable I/O Command Set specification for details.

The controller shall abort the command with a status code of Invalid I/O Command Set if the Command Set Identifier is not associated with an I/O Command Set that supports the Namespace Granularity List.

5.17.2.16 UUID List (CNS 17h)

The format of the UUID List is defined in Figure 284. Each UUID List entry is either 0h, the NVMe Invalid UUID, or a valid UUID. Valid UUIDs are those which are non-zero and are not the NVMe Invalid UUID (refer to section 8.25).

If bit 9 (UUID List) is set to '1' in the Controller Attributes (CTRATT) field in the Identify Controller data structure (refer to Figure 275), then:

- The UUID List shall contain at least one valid UUID (refer to section 8.25);
- The UUID 1 field shall contain a non-zero value; and
- A UUID field cleared to 0h indicates the end of the UUID List.

The list may be in any order.

Figure 284: UUID List

Bytes	Description
31:00	Reserved
63:32	**UUID 1:** This field contains the first UUID List Entry in the list.
95:64	**UUID 2:** This field contains the second UUID List Entry in the list, if present, otherwise cleared to 0h.
…	**…**
4063:4032	**UUID 126:** This field contains the last non-zero UUID List Entry in the list, if present, otherwise cleared to 0h.
4095:4064	**UUID 127:** This field shall be cleared to 0h.

The format of a UUID List Entry is defined in Figure 285.

Figure 285: UUID List Entry

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>00</td><td>UUID Lists Entry Header:
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:2</td><td>Reserved</td></tr>
<tr><td>1:0</td><td>Identifier Association: This field indicates whether the UUID is associated with a vendor.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>00b</td><td>No association reported.</td></tr>
<tr><td>01b</td><td>The UUID is associated with the vendor reported in the PCI Vendor ID field of the Identify Controller data structure (refer to Figure 275).</td></tr>
<tr><td>10b</td><td>The UUID is associated with the vendor reported in the PCI Subsystem Vendor ID field of the Identify Controller data structure.</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table></td></tr>
</table></td></tr>
</table>

Figure 285: UUID List Entry

Bytes	Description
15:01	Reserved
31:16	**UUID:** This field contains a 128-bit Universally Unique Identifier (UUID) as specified in RFC 4122. Refer to section 4.3.6.

5.17.2.17 Domain List (CNS 18h)

Figure 286 defines a Domain List. The data structure is an ordered list by Domain Identifier, starting with the first Domain Identifier that is equal to or greater than the Domain Identifier specified in CDW11.DOMID and is accessible by the controller processing the command. The Domain List describes the attributes for each Domain in the list based on the Domain Attributes Entry in Figure 287.

Figure 286: Domain List

Bytes	Description
00	**Number of Identifiers:** This field indicates the number of Domain Attributes Entries in the list. There are up to 31 entries in the list. A value of 0h indicates that there are no entries in the list.
127:01	Reserved
255:128	**Entry 0:** This field contains the first Domain Attributes Entry in the list, if present.
383:256	**Entry 1:** This field contains the second Domain Attributes Entry in the list, if present.
…	**…**
(N*128+255): (N*128+128)	**Entry N:** This field contains the N+1 Domain Attributes Entry in the list, if present.

Figure 287: Domain Attributes Entry

Bytes	Description
01:00	**Domain Identifier:** This field indicates the identifier of the Domain accessible by the controller that is described by this entry.
15:02	Reserved
31:16	**Total Domain Capacity:** This field indicates the total NVM capacity in this Domain. The value is in bytes.
47:32	**Unallocated Domain Capacity:** This field indicates the unallocated NVM capacity in this Domain. The value is in bytes.
63:48	**Max Endurance Group Domain Capacity:** This field indicates the maximum capacity of a single Endurance Group in this Domain. If this field is cleared to 0h, the NVM subsystem does not report a maximum Endurance Group Domain Capacity value.
127:64	Reserved

5.17.2.18 Endurance Group List (19h)

An Endurance Group List (refer to Figure 288) of up to 2,047 Endurance Group Identifiers in increasing order is returned containing an Endurance Group Identifier greater than or equal to the value specified in the Endurance Group Identifier (CDW11.ENDGID) field. The list contains Endurance Group Identifiers of Endurance Groups that are accessible by the controller processing the command. If the value specified in the Endurance Group Identifier is greater than ENDGIDMAX, then the controller shall complete the command with a status code of Successful Completion and return an Endurance Group List containing no Endurance Group Identifiers.

Figure 288: Endurance Group List

Bytes	Description
01:00	**Number of Identifiers (N):** This field contains the number of Endurance Group Identifiers in the list. There may be up to 2,047 identifiers in the list. If this field is cleared to 0h, then no Endurance Group Identifiers are in the list.

Figure 288: Endurance Group List

Bytes	Description
03:02	**Identifier 0:** This field contains the first Endurance Group Identifier in the list, if any.
05:04	**Identifier 1:** This field contains the second Endurance Group Identifier in the list, if any.
…	…
(N*2+1):(N*2)	**Identifier N-1:** This field contains the last Endurance Group Identifier in the list.

5.17.2.19 I/O Command Set specific Allocated Namespace ID list (CNS 1Ah)

A list of up to 1,024 namespace IDs is returned to the host containing allocated NSIDs in increasing order that are greater than the value specified in the Namespace Identifier (NSID) field of the Identify command and as specified by the Command Set Identifier (CSI) field of the command. Only NSIDs for namespaces associated with the I/O Command Set specified in CSI are returned. For CSI values not supported by the controller the command is aborted with a status code of Invalid Field in Command.

The controller should abort the command with a status code of Invalid Namespace or Format if the NSID field is set to FFFFFFFEh or FFFFFFFFh. The NSID field may be cleared to 0h to retrieve a Namespace List including the namespace starting with NSID of 1h. The data structure returned is a Namespace List (refer to section 4.4.2).

5.17.2.20 I/O Command Set specific Identify Namespace data structure for an Allocated Namespace ID (CNS 1Bh)

An I/O Command Set specific Identify Namespace data structure (refer to section 5.17.2.5) is returned to the host for the namespace specified in the Namespace Identifier (NSID) field if it is an allocated NSID. If the specified namespace is an unallocated NSID, then the controller returns a zero filled data structure.

The specific Identify Namespace data structure that is returned by this command is specified by the Command Set Identifier (CSI) field in the command (refer to Figure 274). If the I/O Command Set associated with the namespace specified by the NSID field does not support the specific Identify Namespace data structure specified by the CSI field, the controller shall abort the command with a status code of Invalid Field in Command.

If the specified namespace is an invalid NSID, then the controller shall abort the command with a status code of Invalid Namespace or Format. If the NSID field is set to FFFFFFFFh, then the controller should abort the command with a status code of Invalid Namespace or Format.

5.17.2.21 Identify I/O Command Set data structure (CNS 1Ch)

The Identify I/O Command Set data structure (refer to Figure 289) is returned to the host for the controller specified in the Controller ID (CNTID) field of the command if the CNTID field does not have a value of FFFFh. If the CNTID field has a value of FFFFh, then the Identify I/O Command Set data structure is returned to the host for the controller processing the command.

This CNS value shall be implemented if CAP.CSS bit 6 is set to ‘1’.

The Identify I/O Command Set data structure consists of an array of I/O Command Set Vectors (refer to Figure 290) that describe the I/O Command Sets that the controller supports and the combination of supported I/O Command Sets that may be simultaneously used. The I/O Command Set Profile Feature value indicates the index of the I/O Command Set Combination that is currently selected (refer to section 5.27.1.21). I/O Command Set Combination 0 has an index value of 0h, I/O Command Set Combination 1 has an index value of 1h, and so on. Only I/O Command Sets that have a bit set to ‘1’ in the I/O Command Set Vector of the I/O Command Set Combination selected by the I/O Command Set Profile Feature value may be used. All other I/O Command Sets are treated as unsupported I/O Command Sets.

Figure 289: Identify I/O Command Set Data Structure

Bytes	Description
7:0	**I/O Command Set Combination 0:** This field contains an I/O Command Set Vector indicating the first I/O Command Set or combination of I/O Command Sets that are simultaneously supported. If only one I/O Command Set is supported, then this field has only one bit set.
15:8	**I/O Command Set Combination 1:** This field contains an I/O Command Set Vector indicating the second I/O Command Set or combination of I/O Command Sets that are simultaneously supported if a second I/O Command Set combination is supported; otherwise, this field is cleared to 0h. If this field is cleared to 0h, then no further I/O Command Set combinations are supported and subsequent I/O Command Set Combinations shall have a value of 0h.
23:16	**I/O Command Set Combination 2:** This field contains an I/O Command Set Vector indicating the third I/O Command Set or combination of I/O Command Sets that are simultaneously supported if a third I/O Command Set combination is supported; otherwise, this field is cleared to 0h. If this field is cleared to 0h, then no further I/O Command Set combinations are supported and subsequent I/O Command Set Combinations shall have a value of 0h.
…	**…**
4095:4088	**I/O Command Set Combination 511:** This field contains an I/O Command Set Vector indicating the 511th I/O Command Set or combination of I/O Command Sets that are simultaneously supported if a 511th I/O Command Set combination is supported; otherwise, this field is cleared to 0h.

Figure 290: I/O Command Set Vector

Bit	Description
63:3	Reserved
2	**Zoned Namespace Command Set:** This bit is set to ‘1’ if the Zoned Namespace Command Set is selected. This bit is cleared to ‘0’ if the Zoned Namespace Command Set is not selected.
1	**Key Value Command Set:** This bit is set to ‘1’ if the Key Value Command Set is selected. This bit is cleared to ‘0’ if the Key Value Command Set is not selected.
0	**NVM Command Set:** This bit is set to ‘1’ if the NVM Command Set is selected. This bit is cleared to ‘0’ if the NVM Command Set is not selected.

5.17.3 Command Completion

Upon completion of the Identify command, the controller posts a completion queue entry to the Admin Completion Queue.

5.18 Keep Alive command

The Keep Alive command (refer to section 5.27.1.12) and associated functionality is used by the host to determine that the controller is operational and used by the controller to determine that the host is operational. The host and controller are operational when each is accessible and able to issue or process commands. The controller indicates the granularity of the Keep Alive Timer in the KAS field in the Identify Controller data structure (refer to Figure 275).

If a Keep Alive Timeout has been enabled on the Admin Queue, the Keep Alive Timer is restarted when:

- a Keep Alive command (refer to section 3.9.1) is processed; or
- at the end of the Keep Alive Timeout (refer to section 3.9.2) when TBKAS is set to '1' and an Admin command or an I/O command is processed during the Keep Alive Timeout Interval.

All command specific fields are reserved.

5.18.1 Command Completion

Upon completion of the Keep Alive command, the controller shall post a completion queue entry to the Admin Completion Queue indicating the status for the command.

5.19 Lockdown command

The Lockdown command is used to control the Command and Feature Lockdown capability (refer to section 8.4) which configures the prohibition or allowance of execution of the specified command or Set Features command targeting a specific Feature Identifier.

After a successful completion of a Lockdown command prohibiting a command or Feature Identifier, all controllers, if applicable, and all management endpoints, if applicable, in the NVM subsystem behave as described in 8.4.

The Lockdown command uses Command Dword 10 (refer to Figure 291) and Command Dword 14 (refer to Figure 292). All other command specific fields are reserved.

Figure 291: Lockdown – Command Dword 10

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:16</td><td>Reserved</td></tr>
<tr><td>15:8</td><td>Opcode or Feature Identifier (OFI): This field specifies the command opcode or Set Features Feature Identifier identified by the Scope field.</td></tr>
<tr><td>07:06</td><td>Reserved</td></tr>
<tr><td>06:05</td><td>Interface (IFC): This field identifies the interfaces affected by this command. The actions of this command apply if a command is received on the specified interfaces.
<table>
<tr><th>Value</th><th>Affected Interfaces</th></tr>
<tr><td>00b</td><td>Admin Submission Queue</td></tr>
<tr><td>01b</td><td>Admin Submission Queue and out-of-band on a Management Endpoint</td></tr>
<tr><td>10b</td><td>Out-of-band on a Management Endpoint</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>04</td><td>Prohibit (PRHBT): This bit specifies whether to prohibit or allow the command opcode or Set Features Feature Identifier specified by this command. If set to '1', then this command prohibits the execution of the command based on other fields specified in Dword 10. If cleared to '0', then this command allows the execution of the command based on other fields specified in Dword 10.</td></tr>
</table>

Bits	Description
03:00	**Scope (SCP):** This field specifies the contents of the Opcode or Feature Identifier field. **Value** / **Opcode or Feature Identifier Definition** 0h: Admin command opcode 1h: Reserved 2h: A Set Features Feature Identifier 3h: Management Interface Command Set opcode (refer to the NVM Express Management Interface Specification) 4h: PCIe Command Set opcode (refer to the NVM Express Management Interface Specification) 5h-Fh: Reserved

If the controller supports selection of a UUID:

a) by the Lockdown command; and
b) by the Set Features command (refer to Figure 316 and section 8.24) and for the vendor specific Feature Identifier specified by the Opcode or Feature Identifier field, if the Scope field is set to 2h,

then Command Dword 14 (refer to Figure 292) is used to specify a UUID Index value.

If the controller does not support selection of a UUID:

a) by the Lockdown command;
b) by the Set Features command; or
c) for the vendor specific feature identifier specified by the Opcode or Feature Identifier field, if the Scope field is set to 2h,

then Command Dword 14 does not specify a UUID Index value. If the Scope field is not set to 2h, then UUID Index field is ignored.

Figure 292: Lockdown – Command Dword 14

Bits	Description
31:07	Reserved
06:00	**UUID Index:** Refer to Figure 477.

If a controller processes this command specifying a command opcode or Feature Identifier that is not supported as being prohibitable, then the command shall be aborted with a status code of Prohibition of Command Execution Not Supported.

If a controller processes this command with the Interface field set to 01b or 10b and the NVM subsystem does not contain a Management Endpoint, then the command shall be aborted with a status code of Invalid Field in Command.

If a controller processes this command with the Interface field set to 00b or 01b and the Scope field is set to 4h, then the command shall be aborted with a status code of Invalid Field in Command.

It is not an error to attempt to prohibit a command or Feature Identifier that is already prohibited from execution or allow a command or Feature Identifier that is already allowed to be executed.

5.19.1 Command Completion

Upon completion of the Lockdown command, the controller posts a completion queue entry to the Admin Completion Queue.

Lockdown command specific status values are defined in Figure 293.

Figure 293: Lockdown – Command Specific Status Values

Value	Description
28h	**Prohibition of Command Execution Not Supported:** The command was aborted due to the specified opcode or Feature Identifier not supporting being prohibited from execution by the command.

5.20 NVMe-MI Receive command

Refer to the NVM Express Management Interface Specification for details on the NVMe-MI Receive command.

5.21 NVMe-MI Send command

Refer to the NVM Express Management Interface Specification for details on the NVMe-MI Send command.

5.22 Namespace Attachment command

The Namespace Attachment command is used to attach and detach controllers from a namespace. The attach and detach operations are persistent across all reset events. Namespace attach and detach operations are persistent across Virtualization Management commands that set a secondary controller offline.

If the Namespace Attachment command is supported, then the Namespace Management command (refer to section 5.23) shall also be supported.

The Namespace Attachment command uses the Data Pointer and Command Dword 10 fields. All other command specific fields are reserved.

The Select field determines the data structure used as part of the command. The data structure is 4,096 bytes in size. The data structure used for Controller Attach and Controller Detach is a Controller List (refer to section 4.4). The controllers that are to be attached or detached, respectively, are described in the data structure.

If the SEL field specifies the Controller Attach value, then

- If the Maximum Domain Namespace Attachments (MAXDNA) field in the Identify Controller data structure (refer to Figure 275) is non-zero, then:
    - For each controller specified in the controller list, if attaching the namespace to that I/O controller causes the sum of the number of namespaces attached to each I/O controller in the Domain to be greater than the value specified in the MAXDNA field, then the controller shall abort the command with a status code of Namespace Attachment Limit Exceeded;

and

- For each I/O controller specified in the controller list, if the Maximum I/O Controller Namespace Attachments (MAXCNA) field in the Identify Controller data structure for that controller is non-zero, then:
    - If attaching the namespace to that I/O controller causes that I/O controller to have the number of attached namespaces to be greater than the value specified in the MAXCNA field, then the controller shall abort the command with a status code of Namespace Attachment Limit Exceeded.

If an attempt is made to attach a namespace to a controller that does not support the corresponding I/O Command Set, then the command shall be aborted with a status code of I/O Command Set Not Supported.

If an attempt is made to attach a namespace to a controller that supports the corresponding I/O Command Set and the corresponding I/O Command Set is not enabled by the I/O Command Set profile feature, then the command shall be aborted with a status code of I/O Command Set Not Enabled.

Figure 294: Namespace Attachment – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 87 for the definition of this field. If using PRPs, this field shall not be a pointer to a PRP List as the data buffer may not cross more than one page boundary.

Figure 295: Namespace Attachment – Command Dword 10

Bits	Description
31:04	Reserved
03:00	**Select (SEL):** This field selects the type of attachment to perform. 0h: Controller Attach 1h: Controller Detach 2h to Fh: Reserved

5.22.1 Command Completion

When the command is completed, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.

Command specific status values associated with the Namespace Attachment command are defined in Figure 296. For failures, the byte offset of the first failing entry is reported in the Command Specific Information field of the Error Information Log Entry. The controller does not process further entries in the Controller List after an error is encountered.

Figure 296: Namespace Attachment – Command Specific Status Values

Value	Description
18h	**Namespace Already Attached:** The controller is already attached to the namespace specified.
19h	**Namespace Is Private:** The controller is not attached to the namespace. The request to attach the controller could not be completed because the namespace is private and is already attached to one controller.
1Ah	**Namespace Not Attached:** The controller is not attached to the namespace. The request to detach the controller could not be completed.
1Ch	**Controller List Invalid:** The controller list provided is invalid or the controller list contains an Administrative controller.
25h	**ANA Attach Failed:** The controller is not attached to the namespace as a result of an ANA condition (e.g., attaching the controller would result in an ANA Persistent Loss state (refer to section 8.1.3.4)).
27h	**Namespace Attachment Limit Exceeded:** Attaching the namespace to a controller causes maximum number of namespace attachments allowed to be exceeded.
29h	**I/O Command Set Not Supported:** The request to attach the controller could not be completed due to the I/O Command Set corresponding to the namespace is not supported by the controller.
2Ah	**I/O Command Set Not Enabled:** The request to attach the controller could not be completed due to the I/O Command Set corresponding to the namespace is restricted by the I/O Command Set profile feature.

5.23 Namespace Management command

The Namespace Management command is used to manage namespaces (refer to section 8.11), including create and delete operations.

Note: The controller continues to execute commands submitted to I/O Submission Queues while this operation is in progress.

If the Namespace Management command is supported, then the Namespace Attachment command (refer to section 5.22) shall also be supported.

Host software uses the Namespace Attachment command to attach or detach a namespace to or from a controller. The create operation does not attach the namespace to a controller. As a side effect of the delete

operation, the namespace is detached from all controllers as the namespace is no longer present in the system. It is recommended that host software detach all controllers from a namespace prior to deleting the namespace. If the namespace is attached to another controller (i.e., a controller other than the controller processing the operation) and that controller has Namespace Attribute Notices enabled (refer to Figure 326), when a delete operation is requested, then as part of the delete operation a Namespace Attribute Notice is issued by that controller to indicate a namespace change.

The data structure used for the create operation is defined in Figure 300 and the CSI field specifies the I/O Command Set specification. There is no data structure transferred for the delete operation.

The Namespace Management command uses the Data Pointer, Command Dword 10, and Command Dword 11 fields. All other command specific fields are reserved.

The Namespace Identifier (NSID) field is used as follows for create and delete operations:

- **Create:** The NSID field is reserved for this operation; host software clears this field to a value of 0h. The controller shall select an available Namespace Identifier to use for the operation; or
- **Delete:** This field specifies the previously created namespace to delete in this operation. Specifying a value of FFFFFFFFh is used to delete all namespaces in the NVM subsystem. If the value of FFFFFFFFh is specified and there are zero valid namespaces, the command completes successfully.

Figure 297: Namespace Management – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 87 for the definition of this field. If using PRPs, this field shall not be a pointer to a PRP List as the data buffer may not cross more than one page boundary.

Figure 298: Namespace Management – Command Dword 10

Bits	Description
31:04	Reserved
03:00	**Select (SEL):** This field selects the type of management operation to perform. Value: 0h – Create; 1h – Delete; 2h to Fh – Reserved

Figure 299: Namespace Management – Command Dword 11

Bits	Description
31:24	**Command Set Identifier (CSI):** For a create operation (i.e., SEL 0h), this field specifies the I/O Command Set for the created namespace. A CSI value of 0h creates a namespace using the NVM Command Set. For all other operations this field is reserved. Values for this field are defined by Figure 274.
23:0	Reserved

Figure 300: Namespace Management – Data Structure for Create

Bytes	Description
511:00	Specific to the I/O Command Set (refer to the Namespace Management command section of the applicable I/O Command Set specification)
1023:512	Reserved
4095:1024	Vendor specific

5.23.1 Command Completion

When the command is completed, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.

Namespace Management command specific status values (i.e., SCT field set to 1h) are shown in Figure 301.

Figure 301: Namespace Management – Command Specific Status Values

Value	Description
0Ah	**Invalid Format:** The User Data Format specified is not supported. This may be due to various conditions, including: 1) specifying an invalid User Data Format number; 2) enabling protection information when there are not sufficient resources (e.g. metadata per LBA); or 3) the specified format is not available in the current configuration.
15h	**Namespace Insufficient Capacity:** Creating the namespace requires more unallocated capacity than is currently available. The Command Specific Information field of the Error Information log page specifies the total amount of unallocated NVM capacity required to create the namespace in bytes.
16h	**Namespace Identifier Unavailable:** The number of namespaces supported has been exceeded.
1Bh	**Thin Provisioning Not Supported:** Thin provisioning is not supported by the controller.
24h	**ANA Group Identifier Invalid:** The specified ANA Group Identifier (ANAGRPID) is not supported in the submitted command. This may be due to various conditions, including: a) specifying an ANAGRPID that does not exist; b) the controller does not allow an ANAGRPID to be specified (i.e., bit 7 in the ANACAP field is cleared to '0'); or c) the specified ANAGRPID is not supported by the controller processing the command (e.g., the specified value exceeds ANAGRPMAX (refer to Figure 275)). If the host specified a non-zero ANAGRPID, retrying the command with the ANAGRPID field cleared to 0h may succeed.
29h	**I/O Command Set Not Supported:** The I/O Command Set specified for a create operation is not supported by the controller.

Dword 0 of the completion queue entry contains the Namespace Identifier created. The definition of Dword 0 of the completion queue entry is in Figure 302.

Figure 302: Namespace Management – Completion Queue Entry Dword 0

Bits	Description
31:00	**Namespace Identifier (NSID):** This field specifies the namespace identifier created in a Create operation. This field is reserved for all other operations.

5.24 Sanitize command

The Sanitize command is used to start a sanitize operation or to recover from a previously failed sanitize operation. The sanitize operation types that may be supported are Block Erase, Crypto Erase, and Overwrite. All sanitize operations are processed in the background (i.e., completion of the Sanitize command does not indicate completion of the sanitize operation). Refer to section 8.21 for details on the sanitize operation.

If the NVM subsystem supports multiple domains and the Sanitize command is not able to start a sanitize operation as a result of the NVM subsystem being divided (refer to section 3.2.4), then the Sanitize command shall be aborted with a status code of Asymmetric Access Inaccessible or Asymmetric Access Persistent Loss.

When a sanitize operation starts on any controller, all controllers in the NVM subsystem:

- Shall clear any outstanding Sanitize Operation Completed asynchronous event or Sanitize Operation Completed With Unexpected Deallocation asynchronous event;
- Shall update the Sanitize Status log (refer to section 5.16.1.25);
- Shall abort any command (submitted or in progress) not allowed during a sanitize operation with a status code of Sanitize In Progress (refer to section 8.21.1);
- Shall abort device self-test operations in progress;
- Suspends autonomous power state management activities as described in section 8.15.2; and
- Shall release stream identifiers for any open streams.

If a sanitize operation is not in progress and the most recent sanitize operation did not fail, then a Sanitize command with a Sanitize Action set to 001b (i.e., Exit Failure Mode) shall complete with a status code of Successful Completion and perform no other action.

While a sanitize operation is in progress, all controllers in the NVM subsystem shall abort any command not allowed during a sanitize operation with a status code of Sanitize In Progress (refer to section 8.21.1) and the Persistent Memory Region shall behave as described in section 8.21.1.

After a sanitize operation fails, all controllers in the NVM subsystem shall abort any command not allowed during a sanitize operation with a status code of Sanitize Failed (refer to section 8.21.1) and the Persistent Memory Region shall behave as described in section 8.21.1 until a subsequent sanitize operation is started or successful recovery from the failed sanitize operation occurs.

If the most recent failed sanitize operation was started in unrestricted completion mode (i.e., the AUSE bit was set to ‘1’ in the Sanitize command), failure recovery requires the host to issue a subsequent Sanitize command in restricted or unrestricted completion mode or to issue a subsequent Sanitize command with the Exit Failure Mode action.

If the most recent failed sanitize operation was started in restricted completion mode (i.e., the AUSE bit was cleared to ‘0’ in the Sanitize command), failure recovery requires the host to issue a subsequent Sanitize command in restricted completion mode. In the case of a sanitize operation failure in restricted completion mode, before starting another sanitize operation:

- any subsequent Sanitize command issued with the Exit Failure Mode action shall be aborted with a status code of Sanitize Failed; and
- any Sanitize command issued in unrestricted completion mode shall be aborted with a status code of Sanitize Failed.

The Sanitize Capabilities field in the Identify Controller data structure indicates:

a) the sanitize operation types supported;
b) whether setting No-Deallocate After Sanitize bit (i.e., Sanitize command Dword 10 bit 9) causes media to be modified after a successful sanitize operation completes; and
c) whether the controller inhibits the functionality of the No-Deallocation After Sanitize bit in the Sanitize command.

If an unsupported sanitize operation type is selected by a Sanitize command, then the controller shall abort the command with a status code of Invalid Field in Command.

If any Persistent Memory Region is enabled in an NVM subsystem, then the controller shall abort any Sanitize command with a status code of Sanitize Prohibited While Persistent Memory Region is Enabled. A sanitize operation is prohibited while any Persistent Memory Region is enabled.

If any namespace is write protected in an NVM subsystem (refer to section 8.12), then the controller aborts any Sanitize command with a status code of Namespace is Write Protected. A sanitize operation is prohibited while any namespace is write protected.

If a firmware activation with reset is pending, then the controller shall abort any Sanitize command.

If the Firmware Commit command that established the pending firmware activation with reset condition returned a status code of:

a) Firmware Activation Requires Controller Level Reset;
b) Firmware Activation Requires Conventional Reset; or
c) Firmware Activation Requires NVM Subsystem Reset,

then the controller should abort the Sanitize command with that same status code.

If the Firmware Commit command that established the pending firmware activation with reset condition completed successfully or returned a status code other than:

a) Firmware Activation Requires Controller Level Reset;
b) Firmware Activation Requires Conventional Reset; or
c) Firmware Activation Requires NVM Subsystem Reset,

then the controller should abort the Sanitize command with a status code of Firmware Activation Requires Controller Level Reset.

Activation of new firmware is prohibited during a sanitize operation (refer to section 8.21.1).

Support for Sanitize commands in a Controller Memory Buffer (i.e., submitted to an Admin Submission Queue in a Controller Memory Buffer or specifying an Admin Completion Queue in a Controller Memory Buffer) is implementation specific. If an implementation does not support Sanitize commands in a Controller Memory Buffer and a controller's Admin Submission Queue or Admin Completion Queue is in the Controller Memory Buffer, then the controller shall abort all Sanitize commands with a status code of Command Not Supported for Queue in CMB.

All sanitize operations (i.e., Block Erase, Crypto Erase, and Overwrite) are performed in the background (i.e., Sanitize command completion does not indicate sanitize operation completion). If a sanitize operation is started, then the controller shall complete the Sanitize command with a status code of Successful Completion. If the controller completes a Sanitize command with any status code other than Successful Completion, then the controller:

- shall not start the sanitize operation for that command;
- shall not modify the Sanitize Status log page; and
- shall not alter any user data.

The Sanitize command uses Command Dword 10 and Command Dword 11. All other command specific fields are reserved.

Figure 303: Sanitize – Command Dword 10

Bits	Description
31:10	Reserved
09	**No-Deallocate After Sanitize:** If set to '1' and the No-Deallocate Inhibited bit (refer to Figure 275) is cleared to '0', then the controller shall not deallocate any user data as a result of successfully completing the sanitize operation. If: a) cleared to '0'; or b) set to '1' and the No-Deallocate Inhibited bit is set to '1', then the controller should deallocate user data as a result of successfully completing the sanitize operation. This bit shall be ignored if the Sanitize Action field is set to 001b (i.e., Exit Failure Mode).
08	**Overwrite Invert Pattern Between Passes (OIPBP):** If set to '1', then the Overwrite Pattern shall be inverted between passes. If cleared to '0', then the overwrite pattern shall not be inverted between passes. This bit shall be ignored unless the Sanitize Action field is set to 011b (i.e., Overwrite).
07:04	**Overwrite Pass Count (OWPASS):** This field specifies the number of overwrite passes (i.e., how many times the media is to be overwritten) using the data from the Overwrite Pattern field of this command. A value of 0h specifies 16 overwrite passes. This field shall be ignored unless the Sanitize Action field is set to 011b (i.e., Overwrite).
03	**Allow Unrestricted Sanitize Exit (AUSE):** If set to '1', then the sanitize operation is performed in unrestricted completion mode. If cleared to '0', then the sanitize operation is performed in restricted completion mode. This bit shall be ignored if the Sanitize Action field is set to 001b (i.e., Exit Failure Mode).

Figure 303: Sanitize – Command Dword 10

Bits	Description
02:00	**Sanitize Action (SANACT):** This field specifies the sanitize action to perform. Value / Description: 000b – Reserved 001b – Exit Failure Mode 010b – Start a Block Erase sanitize operation 011b – Start an Overwrite sanitize operation 100b – Start a Crypto Erase sanitize operation 101b to 111b – Reserved

Figure 304: Sanitize – Command Dword 11

Bits	Description
31:00	**Overwrite Pattern (OVRPAT):** This field is ignored unless the Sanitize Action field in Command Dword 10 is set to 011b (i.e., Overwrite). This field specifies a 32-bit pattern that is used for the Overwrite sanitize operation. Refer to section 8.21.

5.24.1 Command Completion

When the command is complete, the controller shall post a completion queue entry to the Admin Completion Queue indicating the status for the command. All sanitize operations are performed in the background (i.e., completion of the Sanitize command does not indicate completion of the sanitize operation). If a sanitize operation is started, then the Sanitize Status log page shall be updated before posting the completion queue entry for the command that started that sanitize operation.

Sanitize command specific status values (i.e., SCT field set to 1h) are shown in Figure 305.

Figure 305: Sanitize – Command Specific Status Values

Value	Description
0Bh	**Firmware Activation Requires Conventional Reset:** The sanitize operation could not be started because a firmware activation is pending and a Conventional Reset is required.
10h	**Firmware Activation Requires NVM Subsystem Reset:** The sanitize operation could not be started because a firmware activation is pending and an NVM Subsystem Reset is required.
11h	**Firmware Activation Requires Controller Level Reset:** The sanitize operation could not be started because a firmware activation is pending and a Controller Level Reset is required.
23h	**Sanitize Prohibited While Persistent Memory Region is Enabled:** A sanitize operation is prohibited while the Persistent Memory Region is enabled.

5.25 Security Receive command

The Security Receive command transfers the status and data result of one or more Security Send commands that were previously submitted to the controller.

The association between a Security Receive command and previous Security Send commands is dependent on the Security Protocol. The format of the data to be transferred is dependent on the Security Protocol. Refer to SPC-5 for Security Protocol details.

Each Security Receive command returns the appropriate data corresponding to a Security Send command as defined by the rules of the Security Protocol. The Security Receive command data may not be retained if there is a loss of communication between the controller and host, or if a Controller Level Reset occurs.

The fields used are Data Pointer, Command Dword 10, and Command Dword 11 fields. All other command specific fields are reserved.

Figure 306: Security Receive – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 87 for the definition of this field.

Figure 307: Security Receive – Command Dword 10

Bits	Description
31:24	**Security Protocol (SECP):** This field specifies the security protocol as defined in SPC-5. The controller shall abort the command with a status code of Invalid Field in Command if an unsupported value of the Security Protocol is specified.
23:16	**SP Specific 1 (SPSP1):** The value of this field contains bits 15:08 of the Security Protocol Specific field as defined in SPC-5.
15:08	**SP Specific 0 (SPSP0):** The value of this field contains bits 07:00 of the Security Protocol Specific field as defined in SPC-5.
07:00	**NVMe Security Specific Field (NSSF):** Refer to Figure 309 for definition of this field for Security Protocol EAh. For all other Security Protocols this field is reserved.

Figure 308: Security Receive – Command Dword 11

Bits	Description
31:00	**Allocation Length (AL):** The value of this field is specific to the Security Protocol In command with the INC_512 field cleared to 0h as defined in SPC-5.

5.25.1 Command Completion

If the command is completed, then the controller shall post a completion queue entry to the Admin Completion Queue indicating the status for the command.

5.25.2 Security Protocol 00h

A Security Receive command with the Security Protocol field cleared to 00h shall return information about the security protocols supported by the controller. This command is used in the security discovery process and is not associated with a Security Send command. Refer to SPC-5 for the details of Security Protocol 00h and the SP Specific field.

5.25.3 Security Protocol EAh

Security Protocol EAh is assigned for NVMe interface use (refer to ACS-4). This protocol may be used in Security Receive and Security Send commands. The specific usage type is defined by the Security Protocol Specific Field defined in Figure 309.

Figure 309: Security Protocol EAh – Security Protocol Specific Field Values

SP Specific (SPSP) Value	Description	NVMe Security Specific Field (NSSF) Definition
0001h	Replay Protected Memory Block	RPMB Target
0002h to FFFFh	Reserved	Reserved

5.26 Security Send command

The Security Send command is used to transfer security protocol data to the controller. The data structure transferred to the controller as part of this command contains security protocol specific commands to be performed by the controller. The data structure transferred may also contain data or parameters associated with the security protocol commands. Status and data that is to be returned to the host for the security protocol commands submitted by a Security Send command are retrieved with the Security Receive command defined in section 5.25.

The association between a Security Send command and subsequent Security Receive command is Security Protocol field dependent as defined in SPC-5.

The fields used are Data Pointer, Command Dword 10, and Command Dword 11 fields. All other command specific fields are reserved.

Figure 310: Security Send – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 87 for the definition of this field.

Figure 311: Security Send – Command Dword 10

Bits	Description
31:24	**Security Protocol (SECP):** This field specifies the security protocol as defined in SPC-5. The controller shall abort the command with a status code of Invalid Field in Command if a reserved value of the Security Protocol is specified.
23:16	**SP Specific 1 (SPSP1):** The value of this field contains bits 15:08 of the Security Protocol Specific field as defined in SPC-5.
15:08	**SP Specific 0 (SPSP0):** The value of this field contains bits 07:00 of the Security Protocol Specific field as defined in SPC-5.
07:00	**NVMe Security Specific Field (NSSF):** Refer to Figure 309 for definition of this field for Security Protocol EAh. For all other Security Protocols this field is reserved.

Figure 312: Security Send – Command Dword 11

Bits	Description
31:00	**Transfer Length (TL):** The value of this field is specific to the Security Protocol Out command with the INC_512 field cleared to 0h as defined in SPC-5.

5.26.1 Command Completion

If the command is completed, then the controller shall post a completion queue entry to the Admin Completion Queue indicating the status for the command.

5.27 Set Features command

The Set Features command specifies the attributes of the Feature indicated.

The Set Features command uses the Data Pointer, Command Dword 10, and Command Dword 14. The use of Command Dword 11, Command Dword 12, Command Dword 13, and Command Dword 15 fields is Feature specific. If Command Dword 11, Command Dword 12, Command Dword 13, or Command Dword 15 fields are not used, then the Command Dwords are reserved.

Figure 313: Set Features – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 87 for the definition of this field. If using PRPs, this field shall not be a pointer to a PRP List as the data buffer may not cross more than one page boundary. If no data structure is used as part of the specified feature, then this field is not used.

Figure 314: Set Features – Command Dword 10

Bits	Description
31	**Save (SV):** This bit specifies that the controller shall save the attribute so that the attribute persists through all power states and resets. The controller indicates in bit 4 of the Optional NVM Command Support field of the Identify Controller data structure in Figure 275 whether this bit is supported. If the Feature Identifier specified in the Set Features command is not saveable by the controller and the controller receives a Set Features command with the Save bit set to '1', then the command shall be aborted with a status code of Feature Identifier Not Saveable.
30:08	Reserved
07:00	**Feature Identifier (FID):** This field indicates the identifier of the Feature that attributes are being specified for.

If the controller supports selection of a UUID by the Set Features command (refer to Figure 316 and section 8.25) and the controller supports selection of a UUID for the specified vendor specific feature identifier (refer to Figure 316), then Command Dword 14 is used to specify a UUID Index value (refer to Figure 315). If the controller does not support selection of a UUID by the Set Features command or the controller does not support selection of a UUID for the specified vendor specific feature identifier, then Command Dword 14 does not specify a UUID Index value.

Figure 315: Set Features – Command Dword 14

Bits	Description
31:07	Reserved
06:00	**UUID Index:** Refer to Figure 477.

5.27.1 Feature Specific Information

Figure 316 defines the Features that may be configured with a Set Features command and retrieved with a Get Features command. Refer to section 3.1.2 for mandatory, optional, and prohibited features for the various controller types. Some Features utilize a memory buffer to configure or return attributes for a Feature, whereas others only utilize a dword in the command or completion queue entry. If a Feature is not persistent across power cycles and resets, then the current value of that Feature shall be set to the default value of that Feature as part of a Controller Level Reset. For more information on Features, including default value definitions, saved value definitions, and current value definitions, refer to section 4.2.

Upon completion of a Set Features command for a feature, the host should rediscover, re-enumerate and/or re-initialize all capabilities associated with that feature. For example, if a namespace capability change may occur for a feature, then host software should pause the use of any associated namespace, submit the Set Features command for that feature and wait for that command to complete, and then re-issue commands to all namespaces affected by that Set Features command.

There may be commands in execution when a Feature is changed. The new settings may or may not apply to commands already submitted for execution when the Feature is changed. Any commands submitted to a Submission Queue after a Set Features command is successfully completed shall utilize the new settings for the associated Feature. To ensure that a Features values apply to all subsequent commands, the host should allow commands being processed to complete prior to issuing the Set Features command.

If the controller does not support a changeable value for a Feature (e.g., the Feature is not changeable), and a Set Features command for that Feature is processed, then if that command specifies a Feature value that:

- is not the same as the existing value for that Feature, then the controller shall abort that command with a status code of Feature Not Changeable; and
- is the same as the existing value for that Feature, then the controller may:
    - complete that command successfully; or

- abort that command with a status code of Feature Not Changeable.

Figure 316: Set Features – Feature Identifiers

Feature Identifier	Current Setting Persists Across Power Cycle and Reset [2]	Uses Memory Buffer for Attributes	Feature Name
00h	Reserved		
01h	No	No	Arbitration
02h	No	No	Power Management
03h	Refer to the NVM Command Set		
04h	No	No	Temperature Threshold
05h	Refer to the NVM Command Set		
06h	No	No	Volatile Write Cache
07h	No	No	Number of Queues
08h	No	No	Interrupt Coalescing
09h	No	No	Interrupt Vector Configuration
0Ah	Refer to the NVM Command Set		
0Bh	No	No	Asynchronous Event Configuration
0Ch	No	Yes	Autonomous Power State Transition
0Dh	No [3]	No [4]	Host Memory Buffer
0Eh	No	Yes	Timestamp
0Fh	No	No	Keep Alive Timer
10h	Yes	No	Host Controlled Thermal Management
11h	No	No	Non-Operational Power State Config
12h	Yes	No	Read Recovery Level Config
13h	No	Yes	Predictable Latency Mode Config
14h	No	No	Predictable Latency Mode Window
15h	Refer to the NVM Command Set		
16h	No	Yes	Host Behavior Support
17h	Yes	No	Sanitize Config
18h	No	No	Endurance Group Event Configuration
19h	Yes	No	I/O Command Set Profile
1Ah	Yes	No	Spinup Control
1Bh to 1Fh	Reserved		
20h	Refer to the Key Value Command Set		
21h to 77h	Reserved		
78h to 7Ch	Reserved for Management Features.		
7Dh	No	Yes	Enhanced Controller Metadata
7Eh	No	Yes	Controller Metadata
7Fh	No	Yes	Namespace Metadata
80h	Yes	No	Software Progress Marker
81h	No	Yes	Host Identifier
82h	No	No	Reservation Notification Mask
83h	Yes	No	Reservation Persistence
84h	No	No	Namespace Write Protection Config
85h to BFh			Reserved

Figure 316: Set Features – Feature Identifiers

Feature Identifier	Current Setting Persists Across Power Cycle and Reset [2]	Uses Memory Buffer for Attributes	Feature Name
C0h to FFh			Vendor Specific[1, 5]

Notes:
1. The behavior of a controller in response to an inactive namespace ID to a vendor specific Feature Identifier is vendor specific.
2. This column is only valid if the feature is not saveable (refer to section 4.2). If the feature is saveable, then this column is not used.
3. The controller does not save settings for the Host Memory Buffer feature across power states and reset events, however, host software may restore the previous values. Refer to section 8.9.
4. The feature does not use a memory buffer for Set Features commands and does use a memory buffer for Get Features commands. Refer to section 8.9.
5. Selection of a UUID may be supported. Refer to section 8.25.

5.27.1.1 Arbitration (Feature Identifier 01h)

This Feature controls command arbitration. Refer to section 3.4.4 for command arbitration details. The attributes are specified in Command Dword 11.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 317 are returned in Dword 0 of the completion queue entry for that command.

Figure 317: Arbitration & Command Processing – Command Dword 11

Bits	Description
31:24	**High Priority Weight (HPW):** This field defines the number of commands that may be executed from the high priority service class in each arbitration round. This is a 0's based value.
23:16	**Medium Priority Weight (MPW):** This field defines the number of commands that may be executed from the medium priority service class in each arbitration round. This is a 0's based value.
15:08	**Low Priority Weight (LPW):** This field defines the number of commands that may be executed from the low priority service class in each arbitration round. This is a 0's based value.
07:03	Reserved
02:00	**Arbitration Burst (AB):** Indicates the maximum number of commands that the controller may fetch at one time from a particular Submission Queue. The value is expressed as a power of two (e.g., 000b indicates one, 011b indicates eight). A value of 111b indicates no limit.

5.27.1.2 Power Management (Feature Identifier 02h)

This Feature allows the host to configure the power state. The attributes are specified in Command Dword 11 (refer to Figure 318).

Upon successful completion of a Set Features command for this feature, the controller shall be in the Power State specified. For a transition to a non-operational power state, the device may exceed the power indicated for that non-operational power state as defined in section 8.15.1 (e.g., while completing this command). If enabled, autonomous power state transitions continue to occur from the new state.

If a Get Features command is submitted for this Feature, the attributes described in Figure 319 are returned in Dword 0 of the completion queue entry for that command.

Figure 318: Power Management – Command Dword 11

Bits	Description
31:08	Reserved
07:05	**Workload Hint (WH):** This field indicates the type of workload expected. This hint may be used by the NVM subsystem to optimize performance. Refer to section 8.15.3 for more details.

Figure 318: Power Management – Command Dword 11

Bits	Description
04:00	**Power State (PS):** This field indicates the new power state into which the controller is requested to transition. This power state shall be one supported by the controller as indicated in the Number of Power States Supported (NPSS) field in the Identify Controller data structure. If the power state specified is not supported, the controller shall abort the command and should return an error of Invalid Field in Command.

Figure 319: Power Management – Completion Queue Entry Dword 0

Bits	Description
31:08	Reserved
07:05	**Workload Hint (WH):** This field indicates the type of workload. Refer to section 8.15.3 for more details.
04:00	**Power State (PS):** This field indicates the current power state of the controller, or the power state into which the controller is transitioning.

5.27.1.3 Temperature Threshold (Feature Identifier 04h)

A controller may report up to nine temperature values in the SMART / Health Information log page (i.e., the Composite Temperature and Temperature Sensor 1 through Temperature Sensor 8; refer to Figure 207). Associated with each implemented temperature sensor is an over temperature threshold and an under temperature threshold. When a temperature is greater than or equal to its corresponding over temperature threshold or less than or equal to its corresponding under temperature threshold, then bit 1 of the Critical Warning field in the SMART / Health Information log page (refer to section 5.16.1.3) is set to '1'. This may trigger an asynchronous event.

The over temperature threshold feature shall be implemented for Composite Temperature. The under temperature threshold Feature shall be implemented for Composite Temperature if a non-zero Warning Composite Temperature Threshold (WCTEMP) field value is reported in the Identify Controller data structure (refer to Figure 275). The over temperature threshold and under temperature threshold features shall be implemented for all implemented temperature sensors (i.e., all Temperature Sensor fields that report a non-zero value).

The default value of the over temperature threshold feature for Composite Temperature is the value in the Warning Composite Temperature Threshold (WCTEMP) field in the Identify Controller data structure if WCTEMP is non-zero; otherwise, the default value is implementation specific. The default value of the under temperature threshold feature for Composite Temperature is implementation specific. The default value of the over temperature threshold for all implemented temperature sensors is FFFFh. The default value of the under temperature threshold for all implemented temperature sensors is 0h.

If a Get Features command is submitted for this Feature, the temperature threshold selected by Command Dword 11 is returned in Dword 0 of the completion queue entry for that command.

Figure 320: Temperature Threshold – Command Dword 11

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:22</td><td>Reserved</td></tr>
<tr><td>21:20</td><td>Threshold Type Select (THSEL): This field selects the threshold type that is modified by a Set Features command and whose threshold value is returned by a Get Features command.
<table><tr><th>Value</th><th>Description</th></tr><tr><td>00b</td><td>Over Temperature Threshold</td></tr><tr><td>01b</td><td>Under Temperature Threshold</td></tr><tr><td>10b to 11b</td><td>Reserved</td></tr></table></td></tr>
</table>

Figure 320: Temperature Threshold – Command Dword 11

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>19:16</td><td>Threshold Temperature Select (TMPSEL): This field selects the temperature whose threshold is modified by a Set Features command and whose threshold value is returned by a Get Features command.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>Composite Temperature</td></tr>
<tr><td>1h</td><td>Temperature Sensor 1</td></tr>
<tr><td>2h</td><td>Temperature Sensor 2</td></tr>
<tr><td>3h</td><td>Temperature Sensor 3</td></tr>
<tr><td>4h</td><td>Temperature Sensor 4</td></tr>
<tr><td>5h</td><td>Temperature Sensor 5</td></tr>
<tr><td>6h</td><td>Temperature Sensor 6</td></tr>
<tr><td>7h</td><td>Temperature Sensor 7</td></tr>
<tr><td>8h</td><td>Temperature Sensor 8</td></tr>
<tr><td>9h to Eh</td><td>Reserved</td></tr>
<tr><td>Fh</td><td>All implemented temperature sensors in a Set Features command. Reserved in a Get Features command.</td></tr>
</table></td></tr>
<tr><td>15:00</td><td>Temperature Threshold (TMPTH): Indicates the threshold value for the temperature sensor and threshold type specified in Kelvins.</td></tr>
</table>

5.27.1.4 Volatile Write Cache (Feature Identifier 06h)

This Feature controls the volatile write cache, if present, on the controller. If a volatile write cache is present (refer to the VWC field in Figure 275), then this feature shall be supported. The attributes are specified in Command Dword 11.

Note: If the controller is able to guarantee that data present in a write cache is written to non-volatile media on loss of power, then that write cache is considered non-volatile and this feature does not apply to that write cache.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 321 are returned in Dword 0 of the completion queue entry for that command.

If a volatile write cache is not present, then a Set Features command specifying the Volatile Write Cache feature identifier shall abort with a status code of Invalid Field in Command, and a Get Features command specifying the Volatile Write Cache feature identifier should abort with a status code of Invalid Field in Command.

Figure 321: Volatile Write Cache – Command Dword 11

Bits	Description
31:01	Reserved
00	**Volatile Write Cache Enable (WCE):** If set to ‘1’, then the volatile write cache is enabled. If cleared to ‘0’, then the volatile write cache is disabled.

5.27.1.5 Number of Queues (Feature Identifier 07h)

This Feature indicates the number of queues that the host requests for the controller processing the command. This feature shall only be issued during initialization prior to creation of any I/O Submission and/or Completion Queues. If a Set Features command is issued for this feature after creation of any I/O Submission and/or I/O Completion Queues, then the Set Features command shall abort with status code of Command Sequence Error. The controller shall not change the value allocated between resets. For a Set Features command, the attributes are specified in Command Dword 11 (refer to Figure 322). For a Get Features command, Dword 11 is ignored.

If a Set Features or Get Features command is submitted for this Feature, the attributes specified in Figure 323 are returned in Dword 0 of the completion queue entry for that command.

Figure 322: Number of Queues – Command Dword 11

Bits	Description
31:16	**Number of I/O Completion Queues Requested (NCQR):** Indicates the number of I/O Completion Queues requested by software. This number does not include the Admin Completion Queue. A minimum of one queue shall be requested, reflecting that the minimum support is for one I/O Completion Queue. This is a 0's based value. The maximum value that may be specified is 65,534 (i.e., 65,535 I/O Completion Queues). If the value specified is 65,535, the controller should abort the command with a status code of Invalid Field in Command.
15:00	**Number of I/O Submission Queues Requested (NSQR):** Indicates the number of I/O Submission Queues requested by software. This number does not include the Admin Submission Queue. A minimum of one queue shall be requested, reflecting that the minimum support is for one I/O Submission Queue. This is a 0's based value. The maximum value that may be specified is 65,534 (i.e., 65,535 I/O Submission Queues). If the value specified is 65,535, the controller should abort the command with a status code of Invalid Field in Command.

Note: The value allocated may be smaller or larger than the number of queues requested, often in virtualized implementations. The controller may not have as many queues to allocate as are requested. Alternatively, the controller may have an allocation unit of queues (e.g., power of two) and may supply more queues to host software to satisfy its allocation unit.

Figure 323: Number of Queues – Completion Queue Entry Dword 0

Bits	Description
31:16	**Number of I/O Completion Queues Allocated (NCQA):** Indicates the number of I/O Completion Queues allocated by the controller. A minimum of one queue shall be allocated, reflecting that the minimum support is for one I/O Completion Queue. The value may not match the number requested by host software. This is a 0's based value.
15:00	**Number of I/O Submission Queues Allocated (NSQA):** Indicates the number of I/O Submission Queues allocated by the controller. A minimum of one queue shall be allocated, reflecting that the minimum support is for one I/O Submission Queue. The value may not match the number requested by host software. This is a 0's based value.

5.27.1.6 Interrupt Coalescing (Feature Identifier 08h)

This Feature configures interrupt coalescing settings. The controller should signal an interrupt when either the Aggregation Time or the Aggregation Threshold conditions are met. If either the Aggregation Time or the Aggregation Threshold fields are cleared to 0h, then an interrupt may be generated (i.e., interrupt coalescing is implicitly disabled). This Feature applies only to the I/O Queues. It is recommended that interrupts for commands that complete in error are not coalesced. The settings are specified in Command Dword 11.

If the controller detects that interrupts are already being processed for this vector, then the controller may delay additional interrupts. Specifically, if the Completion Queue Head Doorbell property is being updated that is associated with a particular interrupt vector, then the controller has a positive indication that completion queue entries are already being processed. In this case, the aggregation time and/or the aggregation threshold may be reset/restarted upon the associated property write. This may result in interrupts being delayed indefinitely in certain workloads where the aggregation time or aggregation threshold is non-zero.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 324 are returned in Dword 0 of the completion queue entry for that command.

This Feature is valid when the controller is configured for Pin Based, MSI, Multiple MSI or MSI-X interrupts. There is no requirement for the controller to persist these settings if interrupt modes are changed. It is recommended that the host re-issue this Feature after changing interrupt modes.

Figure 324: Interrupt Coalescing – Command Dword 11

Bits	Description
31:16	Reserved
15:08	**Aggregation Time (TIME):** Specifies the recommended maximum time in 100 microsecond increments that a controller may delay an interrupt due to interrupt coalescing. A value of 0h corresponds to no delay. The controller may apply this time per interrupt vector or across all interrupt vectors. The reset value of this setting is 0h.
07:00	**Aggregation Threshold (THR):** Specifies the recommended minimum number of completion queue entries to aggregate per interrupt vector before signaling an interrupt to the host. This is a 0's based value. The reset value of this setting is 0h.

5.27.1.7 Interrupt Vector Configuration (Feature Identifier 09h)

This Feature configures settings specific to a particular interrupt vector. The settings are specified in Command Dword 11.

By default, coalescing settings are enabled for each interrupt vector. Interrupt coalescing is not supported for the Admin Completion Queue.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 325 are returned in Dword 0 of the completion queue entry for that command for the Interrupt Vector specified in Command Dword 11.

Prior to issuing a Set Features command that specifies this Feature, the host shall configure the specified Interrupt Vector with an I/O Completion Queue (refer to section 5.4). If the specified Interrupt Vector is invalid, or not associated with an existing I/O Completion Queue (refer to Figure 157), then the controller should abort the command with a status code of Invalid Field in Command.

Figure 325: Interrupt Vector Configuration – Command Dword 11

Bits	Description
31:17	Reserved
16	**Coalescing Disable (CD):** If set to '1', then any interrupt coalescing settings shall not be applied for this interrupt vector. If cleared to '0', then interrupt coalescing settings apply for this interrupt vector.
15:00	**Interrupt Vector (IV):** This field specifies the interrupt vector for which the configuration settings are applied.

5.27.1.8 Asynchronous Event Configuration (Feature Identifier 0Bh)

This Feature controls the events that trigger an asynchronous event notification to the host. This Feature may be used to disable reporting events in the case of a persistent condition (refer to section 5.2). If the condition for an event is true when the corresponding notice is enabled, then an event is sent to the host. The attributes are specified in Command Dword 11.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 326 are returned in Dword 0 of the completion queue entry for that command.

Figure 326: Asynchronous Event Configuration – Command Dword 11

Bits	Description
31	**Discovery Log Page Change Notification:** This bit indicates that the Discovery controller reports Discovery Log Page Change Notifications. If set to '1', the Discovery controller shall send a notification if Discovery Log Page changes occur.
30:28	Reserved
27	**Zone Descriptor Changed Notices**[2]**:** I/O Command Set specific definition.
26:16	Reserved

Figure 326: Asynchronous Event Configuration – Command Dword 11

Bits	Description
15	**Normal NVM Subsystem Shutdown:** This bit determines whether an asynchronous event notification is sent to the host when the NVM subsystem has started performing a normal shutdown due to an NVM Subsystem Shutdown (refer to Figure 149). If this bit is set to '1', then the Normal NVM Subsystem Shutdown event is sent to the host if an outstanding Asynchronous Event Request command exists at the time this condition occurs. If this bit is cleared to '0', then the controller shall not send the Normal NVM Subsystem Shutdown event to the host.
14	**Endurance Group Event Aggregate Log Change Notices:** This bit determines whether an asynchronous event notification is sent to the host when an event entry for an Endurance Group (refer to section 3.2.3) has been added to the Endurance Group Event Aggregate log (refer to section 5.16.1.15). If this bit is set to '1', then the Endurance Group Event Aggregate Log Change event is sent to the host when this condition occurs. If this bit is cleared to '0', then the controller shall not send the Endurance Group Event Aggregate Log Change event to the host. If Endurance Groups are not supported and this bit is set to '1', then the Set Features command shall be aborted with a status of Invalid Field in Command.
13	**LBA Status Information Alert Notices**[1]**:** I/O Command Set specific definition.
12	**Predictable Latency Event Aggregate Log Change Notices:** This bit determines whether an asynchronous event notification is sent to the host when an event pending entry for an NVM Set (refer to section 5.16.1.12) has been added to the Predictable Latency Event Aggregate Log. If this bit is set to '1', then the Predictable Latency Event Aggregate Log Change event is sent to the host when this condition occurs. If this bit is cleared to '0', then the controller shall not send the Predictable Latency Event Aggregate Log Change event to the host.
11	**Asymmetric Namespace Access Change Notices:** This bit determines whether an asynchronous event notification is sent to the host when an asymmetric namespace access change occurs (i.e., the contents of the Asymmetric Namespace Access log page (refer to section 5.16.1.13) change). If this bit is set to '1', then the Asymmetric Namespace Access Change Notices event is sent to the host when this condition occurs. If this bit is cleared to '0', then the controller shall not send the Asymmetric Namespace Access Change Notices event to the host.
10	**Telemetry Log Notices:** This bit determines whether an asynchronous event notification is sent to the host when the Telemetry Controller-Initiated Data Available field transitions from 0h to 1h in the Telemetry Controller-Initiated log page. If this bit is set to '1', then the Telemetry Log Changed event is sent to the host when this condition occurs. If this bit is cleared to '0', then the controller shall not send the Telemetry Log Changed event to the host.
09	**Firmware Activation Notices:** This bit determines whether an asynchronous event notification is sent to the host for a Firmware Activation Starting event (refer to Figure 147). If this bit is set to '1', then the Firmware Activation Starting event is sent to the host when this condition occurs. If this bit is cleared to '0', then the controller shall not send the Firmware Activation Starting event to the host.
08	**Namespace Attribute Notices:** This bit determines whether an asynchronous event notification is sent to the host for a Namespace Attribute change (refer to Figure 147). If this bit is set to '1', then the Namespace Attribute Changed event is sent to the host when this condition occurs. If this bit is cleared to '0', then the controller shall not send the Namespace Attribute Changed event to the host.
07:00	**SMART / Health Critical Warnings:** This field determines whether an asynchronous event notification is sent to the host for the corresponding Critical Warning specified in the SMART / Health Information log (refer to Figure 207). If a bit is set to '1', then an asynchronous event notification is sent when the corresponding critical warning bit is set to '1' in the SMART / Health Information log. If a bit is cleared to '0', then an asynchronous event notification is not sent when the corresponding critical warning bit is set to '1' in the SMART / Health Information log.

NOTE:
1. Refer to the NVM Command Set Specification.
2. Refer to the Zoned Namespace Command Set Specification.

5.27.1.9 Autonomous Power State Transition (Feature Identifier 0Ch)

This feature configures the settings for autonomous power state transitions, refer to section 8.15.2.

The Autonomous Power State Transition uses Command Dword 11 and specifies the attribute information in the data structure indicated in Figure 327 and the Autonomous Power State Transition data structure consisting of 32 of the entries defined in Figure 328.

If a Get Features command is issued for this Feature, the attributes specified in Figure 327 are returned in Dword 0 of the completion queue entry and the Autonomous Power State Transition data structure, whose entry structure is defined in Figure 328, is returned in the data buffer for that command.

Figure 327: Autonomous Power State Transition – Command Dword 11

Bits	Description
31:01	Reserved
00	**Autonomous Power State Transition Enable (APSTE):** This bit specifies whether autonomous power state transition is enabled. If this bit is set to '1', then autonomous power state transitions are enabled. If this bit is cleared to '0', then autonomous power state transitions are disabled. This bit is cleared to '0' by default.

Each entry in the Autonomous Power State Transition data structure is defined in Figure 328. Each entry is 64 bits in size. There is an entry for each of the allowable 32 power states. For power states that are not supported, the unused Autonomous Power State Transition data structure entries shall be cleared to all zeroes. The entries begin with power state 0 and then increase sequentially (i.e., power state 0 is described in bytes 7:0, power state 1 is described in bytes 15:8, etc.). The data structure is 256 bytes in size and shall be physically contiguous.

Figure 328: Autonomous Power State Transition – Data Structure Entry

Bits	Description
63:32	Reserved
31:08	**Idle Time Prior to Transition (ITPT):** This field specifies the amount of idle time that occurs in this power state prior to transitioning to the Idle Transition Power State. The time is specified in milliseconds. A value of 0h disables the autonomous power state transition feature for this power state.
07:03	**Idle Transition Power State (ITPS):** This field specifies the power state to which the controller autonomously transitions, after there is a continuous period of idle time in the current power state that exceeds the time specified in the Idle Time Prior to Transition (ITPT) field. If the ITPT field is set to a non-zero value, then the state specified in this field shall be a non-operational state as described in Figure 276. This field should not specify a power state with higher reported idle power than the current power state. If the ITPT field is cleared to 0h, then this field should be cleared to 0h.
02:00	Reserved

The Autonomous Power State Transition feature may interact with the Non-Operational Power State Config feature (refer to section 5.27.1.14). Figure 329 shows these interactions.

Figure 329: Interactions between APSTE and NOPPME

APSTE[1]	NOPPME[2]	Non-operational power state entry	Background operations during non-operational power states
1	1	Entered by host request[3] or by ITPT idle timer[4]	Allowed
0	1	Entered by host request[3]	Allowed
1	0	Entered by host request[3] or by ITPT idle timer[4]	Not allowed
0	0	Entered by host request[3]	Not allowed

Notes:
1. Defined in Figure 327.
2. Defined in Figure 343.
3. Refer to section 5.27.1.2.
4. Refer to Figure 328.

5.27.1.10 Host Memory Buffer (Feature Identifier 0Dh)

This Feature controls the Host Memory Buffer. The attributes are specified in Command Dword 11, Command Dword 12, Command Dword 13, Command Dword 14, and Command Dword 15.

The Host Memory Buffer feature provides a mechanism for the host to allocate a portion of host memory for the exclusive use of the controller. After a successful completion of a Set Features command enabling the host memory buffer, the host shall not write to:

a) The Host Memory Descriptor List (refer to Figure 335); and
b) the associated host memory region (i.e., the memory regions described by the Host Memory Descriptor List),

until the host memory buffer has been disabled.

If the host memory buffer is enabled, then a Set Features command to enable the host memory buffer (i.e., the EHM bit (refer to Figure 330) set to '1') shall abort with a status code of Command Sequence Error.

If the host memory buffer is not enabled, then a Set Features command to disable the host memory buffer (i.e., the EHM bit (refer to Figure 330) cleared to '0') shall succeed without taking any action.

After a successful completion of a Set Features command that disables the host memory buffer, the controller shall not access any data in the host memory buffer until the host memory buffer has been enabled. The controller should retrieve any necessary data from the host memory buffer in use before posting the completion queue entry for the Set Features command that disables the host memory buffer. Posting of the completion queue entry for the Set Features command that disables the host memory buffer acknowledges that it is safe for the host software to modify the host memory buffer contents. Refer to section 8.9.

Figure 330: Host Memory Buffer – Command Dword 11

Bits	Description
31:02	Reserved
01	**Memory Return (MR):** If set to '1', then the host is returning memory previously allocated to the controller for use as the host memory buffer (HMB). That memory may have been in use for the HMB prior to a reset or entering the Runtime D3 state (e.g., prior to the HMB being disabled). A returned host memory buffer shall have the exact same size, descriptor list address, descriptor list contents, and host memory buffer contents as last seen by the controller before the host memory buffer was disabled (i.e., a Set Features command with the EHM bit cleared to '0' was processed). If cleared to '0', then the host is allocating host memory resources with undefined content.
00	**Enable Host Memory (EHM):** If set to '1', then the host memory buffer shall be enabled and the controller may use the host memory buffer. If cleared to '0', then the host memory buffer shall be disabled, and the controller shall not use the host memory buffer. If a Set Features command is processed with this bit cleared to '0', then the controller shall ignore Command Dword 12, Command Dword 13, Command Dword 14, and Command Dword 15.

Figure 331: Host Memory Buffer – Command Dword 12

Bits	Description
31:00	**Host Memory Buffer Size (HSIZE):** This field specifies the size of the host memory buffer allocated in memory page size (CC.MPS) units.

Figure 332: Host Memory Buffer– Command Dword 13

Bits	Description
31:00	**Host Memory Descriptor List Lower Address (HMDLLA):** This field specifies the least significant 32 bits of the physical location of the Host Memory Descriptor List (refer to Figure 335) for the Host Memory Buffer. This address shall be 16 byte aligned, indicated by bits 3:0 being cleared to 0h. NOTE: The controller shall operate as if bits 3:0 are cleared to 0h. However, the controller is not required to check that bits 3:0 are cleared to 0h.

Figure 333: Host Memory Buffer – Command Dword 14

Bits	Description
31:00	**Host Memory Descriptor List Upper Address (HMDLUA):** This field specifies the most significant 32 bits of the physical location of the Host Memory Descriptor List for the Host Memory Buffer.

The Host Memory Descriptor List Address (HMDLLA/HMDLUA) specifies the address of a physically contiguous data structure in host memory that describes the address and length pairs of the Host Memory Buffer. The number of address and length pairs is specified in the Host Memory Descriptor List Entry Count in Figure 334. The Host Memory Descriptor List is described in Figure 335.

Figure 334: Host Memory Buffer – Command Dword 15

Bits	Description
31:00	**Host Memory Descriptor List Entry Count (HMDLEC):** This field specifies the number of entries provided in the Host Memory Descriptor List.

Figure 335: Host Memory Buffer – Host Memory Descriptor List

Bytes	Description
15:0	Host Memory Buffer Descriptor (refer to Figure 336) Entry 0
31:16	Host Memory Buffer Descriptor Entry 1
47:32	Host Memory Buffer Descriptor Entry 2
63:48	Host Memory Buffer Descriptor Entry 3
…	
16*n+15:16*n	Host Memory Buffer Descriptor Entry n (where n = HMDLEC - 1 (refer to Figure 334))

Each Host Memory Buffer Descriptor Entry shall describe a host memory address in memory page size units and the number of contiguous memory page size units associated with the host address.

Figure 336: Host Memory Buffer – Host Memory Buffer Descriptor Entry

Bits	Description
127:96	Reserved
95:64	**Buffer Size (BSIZE):** Indicates the number of contiguous memory page size (CC.MPS) units for this descriptor.
63:00	**Buffer Address (BADD):** Indicates the host memory address for this descriptor aligned to the memory page size (CC.MPS). The least significant bits (*n*:0) of this field indicate the offset within the memory page is 0h (e.g., if the memory page size is 4 KiB, then bits 11:00 shall be 0h; if the memory page size is 8 KiB, then bits 12:00 shall be 0h).

If a Get Features command is issued for this Feature, the attributes specified in Figure 337 are returned in Dword 0 of the completion queue entry and the Host Memory Buffer Attributes data structure, whose structure is defined in Figure 338, is returned in the data buffer for that command.

Figure 337: Host Memory Buffer – Completion Queue Entry Dword 0

Bits	Description
31:01	Reserved
00	**Enable Host Memory (EHM):** If set to '1', then the host memory buffer is enabled and the controller may use the host memory buffer. If cleared to '0', then the host memory buffer is disabled, and the controller is not using the host memory buffer.

Figure 338: Host Memory Buffer – Attributes Data Structure

Bytes	Description
3:0	**Host Memory Buffer Size (HSIZE):** This field indicates the size of the host memory buffer allocated in memory page size units.
7:4	**Host Memory Descriptor List Address Lower (HMDLAL):** This field indicates the least significant 32 bits of the physical location of the Host Memory Descriptor List (refer to Figure 335) for the host memory buffer. This address shall be 16 byte aligned. The least significant 4 bits shall be cleared to '0'.
11:8	**Host Memory Descriptor List Address Upper (HMDLAU):** This field indicates the most significant 32 bits of the physical location of the Host Memory Descriptor List (refer to Figure 335) for the host memory buffer.
15:12	**Host Memory Descriptor List Entry Count (HMDLEC):** This field indicates the number of valid Host Memory Descriptor Entries (refer to Figure 336) in the Host Memory Descriptor List (refer to Figure 335).
4095:16	Reserved

5.27.1.11 Timestamp (Feature Identifier 0Eh)

The Feature enables the host to set a timestamp value in the controller. A controller indicates support for the Timestamp feature through the Optional NVM Command Support (ONCS) field in the Identify Controller data structure. The Timestamp field value (refer to Figure 339) in a Set Features command sets a timestamp value in the controller. After the current value for this Feature is set, the controller updates that value as time passes. A Get Features command that requests the current value reports the timestamp value in the controller at the time the Get Features command is processed (e.g., the value set with a Set Features command for the current value plus the elapsed time since being set).

Note: If the Timestamp feature is saveable (refer to Figure 195) and the host saves a value, then the timestamp value restored after a subsequent power on or reset event is the value that was saved (refer to section 4.2). As a result, the timestamp may appear to move backwards in time.

The accuracy of a Timestamp value after initialization may be affected by vendor specific factors, such as whether the controller continuously counts after the timestamp is initialized, or whether the controller stops counting during certain intervals (e.g., non-operational power states). If the controller stops counting during such intervals, then the Synch bit in the Timestamp – Data Structure for Get Features (refer to Figure 340) shall be set to '1'.

If the controller maintains (i.e., continues to update) the timestamp value across any type of Controller Level Reset (e.g., across a Controller Reset), then the controller shall also preserve the Timestamp Origin field (refer to Figure 340) across that type of Controller Level Reset.

If the controller does not maintain the value of the timestamp across the most recent Controller Level Reset, then the Timestamp field is cleared to 0h due to that Controller Level Reset.

Timestamp values should not be used for security applications. Other application use of the Timestamp feature is outside the scope of this specification.

If a Set Features command is issued for this Feature, the data structure specified in Figure 339 is transferred in the data buffer for that command, specifying the Timestamp value.

Figure 339: Timestamp – Data Structure for Set Features

Bytes	Description
05:00	**Timestamp:** Number of milliseconds that have elapsed since midnight, 01-Jan-1970, UTC.
07:06	Reserved

If a Get Features command is issued for this Feature, the data structure specified in Figure 340 is returned in the data buffer for that command.

Figure 340: Timestamp – Data Structure for Get Features

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>05:00</td><td>Timestamp:
If the Timestamp Origin field cleared to 000b, then this field is set to the time in milliseconds since the last Controller Level Reset.
If the Timestamp Origin field is set to 001b, then this field is set to the last Timestamp value set by the host, plus the time in milliseconds since the Timestamp was set. If the sum of the Timestamp value set by the host and the elapsed time exceeds 2^48, the value returned should be reduced modulo 2^48.
If the Synch bit is set to ‘1’, then the Timestamp value may be reduced by vendor specific time intervals not counted by the controller.</td></tr>
<tr><td>06</td><td>
<table>
<tr><th>Bits</th><th>Attribute</th><th>Definition</th></tr>
<tr><td>07:04</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>03:01</td><td>Timestamp Origin</td><td>
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>The Timestamp field was initialized to 0h by a Controller Level Reset.</td></tr>
<tr><td>001b</td><td>The Timestamp field was initialized with a Timestamp value using a Set Features command.</td></tr>
<tr><td>010b to 111b</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>00</td><td>Synch</td><td>
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0</td><td>The controller counted time in milliseconds continuously since the Timestamp value was initialized.</td></tr>
<tr><td>1</td><td>The controller may have stopped counting during vendor specific intervals after the Timestamp value was initialized (e.g., non-operational power states).</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>07</td><td>Reserved</td></tr>
</table>

5.27.1.12 Keep Alive Timer (Feature Identifier 0Fh)

This Feature controls the Keep Alive Timer. Refer to section 3.9 for Keep Alive details. The attributes are specified in Command Dword 11.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 341 are returned in Dword 0 of the completion queue entry for that command.

Figure 341: Keep Alive Timer – Command Dword 11

Bits	Description
31:00	**Keep Alive Timeout (KATO):** This field specifies the timeout value for the Keep Alive feature in milliseconds. The controller rounds up the value specified to the granularity indicated in the KAS field in the Identify Controller data structure. If cleared to 0h, then the Keep Alive Timer is disabled. The default value for this field is 0h for NVMe transports that do not require use of the Keep Alive feature (e.g., NVMe over PCIe). For NVMe transports that require use of the Keep Alive feature (e.g., RDMA and TCP), the default value for this field is 1D4C0h (i.e., 120,000 milliseconds or 2 minutes) rounded up to the granularity indicated in the KAS field. Refer to the applicable NVMe Transport Binding specification for details.

5.27.1.13 Host Controlled Thermal Management (Feature Identifier 10h)

This feature configures the settings for the host controlled thermal management feature, refer to section 8.15.5. The host controlled thermal management feature uses Command Dword 11 with the attributes shown in Figure 342.

If a Get Features command is submitted for this feature, then the attributes shown in Figure 342 are returned in Dword 0 of the completion queue entry for that command.

This feature is not namespace specific.

Figure 342: HCTM – Command Dword 11

Bits	Description
31:16	**Thermal Management Temperature 1 (TMT1):** This field specifies the temperature, in Kelvins, when the controller begins to transition to lower power active power states or performs vendor specific thermal management actions while minimizing the impact on performance (e.g., light throttling) in order to attempt to reduce the Composite Temperature. A value cleared to 0h, specifies that this part of the Feature shall be disabled. The range of values that are supported by the controller are indicated in the Minimum Thermal Management Temperature field and Maximum Thermal Management Temperature field in the Identify Controller data structure in Figure 275. If the host attempts to set this field to a value less than the value contained in the Minimum Thermal Management Temperature field or greater than the value contained in the Maximum Thermal Management Temperature field in the Identify Controller data structure in Figure 275, then the command shall abort with a status code of Invalid Field in Command. If the host attempts to set this field to a value greater than or equal to the value contained in the Thermal Management Temperature 2 field, if non-zero, then the command shall abort with a status code of Invalid Field in Command.

Figure 342: HCTM – Command Dword 11

Bits	Description
15:00	**Thermal Management Temperature 2 (TMT2):** This field specifies the temperature, in Kelvins, when the controller begins to transition to lower power active power states or perform vendor specific thermal management actions regardless of the impact on performance (e.g., heavy throttling) in order to attempt to reduce the Composite Temperature. A value cleared to 0h, specifies that this part of the Feature shall be disabled. The range of values that are supported by the controller are indicated in the Minimum Thermal Management Temperature field and Maximum Thermal Management Temperature field in the Identify Controller data structure in Figure 275. If the host attempts to set this field to a value less than the value contained in the Minimum Thermal Management Temperature field or greater than the value contained in the Maximum Thermal Management Temperature field in the Identify Controller data structure in Figure 275, then the command shall abort with a status code of Invalid Field in Command. If the host attempts to set this field to a non-zero value less than or equal to the value contained in the Thermal Management Temperature 1 field, then the command shall abort with a status code of Invalid Field in Command.

5.27.1.14 Non-Operational Power State Config (Feature Identifier 11h)

This Feature configures non-operational power state settings for the controller. The settings are specified in Command Dword 11.

If a Get Features command is submitted for this Feature, the values in Figure 343 are returned in Dword 0 of the completion queue entry for that command.

Figure 343: Non-Operational Power State Config – Command Dword 11

Bits	Description
31:01	Reserved
00	**Non-Operational Power State Permissive Mode Enable (NOPPME):** If this bit is set to '1', then the controller may temporarily exceed the power limits of any non-operational power state, up to the limits of the last operational power state, to run controller initiated background operations in that state (i.e., Non-Operational Power State Permissive Mode is enabled). If this bit is cleared to '0', then the controller shall not exceed the limits of any non-operational state while running controller initiated background operations in that state (i.e., Non-Operational Power State Permissive Mode is disabled). If Non-Operational Power State Permissive Mode is disabled, then: a) thermal management that requires power (e.g., cooling fans) may be disabled; and b) performance after resuming from the non-operational power state may be degraded until background activity that was not allowed while in that non-operational power state has completed. If the host attempts to set this bit to '1' and the controller does not support Non-Operational Power State Permissive Mode as indicated in the Controller Attributes (CTRATT) field of the Identify Controller data structure, then the controller shall abort the command with a status code of Invalid Field in Command.

The Non-Operational Power State Config feature may interact with the Autonomous Power State Transition feature (refer to section 5.27.1.9). Figure 329 shows these interactions.

5.27.1.15 Read Recovery Level Config (Feature Identifier 12h)

This Feature is used to configure the Read Recovery Level (refer to section 8.17). The attributes are specified in Command Dword 11 and Command Dword 12. Modifying the Read Recovery Level has no effect on the data contained in any associated namespace.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 345 are returned in Dword 0 of the completion queue entry for that command.

Figure 344: Read Recovery Level Config – Command Dword 11

Bits	Description
31:16	Reserved
15:00	**NVM Set Identifier (NVMSETID):** This field specifies the NVM Set to be modified. If NVM Sets are not supported, then this field is ignored and the command applies to all namespaces in the NVM subsystem.

Figure 345: Read Recovery Level Config – Command Dword 12

Bits	Description
31:04	Reserved
03:00	**Read Recovery Level (RRL):** This field sets the Read Recovery Level for the NVM Set specified.

5.27.1.16 Predictable Latency Mode Config (Feature Identifier 13h)

This Feature configures an NVM Set to use Predictable Latency Mode, including warning event thresholds. Predictable Latency Mode and events are disabled by default. The attributes are specified in Command Dword 11, Command Dword 12, and the Deterministic Threshold Configuration data structure.

The NVM Set has transitioned to Predictable Latency Mode when the controller completes a Set Features command successfully with the Predictable Latency Enable bit in Command Dword 12 set to '1'. A transition to the Predictable Latency Mode may be delayed (i.e., the Set Features command completion is delayed) if the NVM subsystem needs to perform background operations on the NVM in order to operate in Predictable Latency Mode. Upon successful completion of this command, the controller shall be in the Non-Deterministic Window.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 347 are returned in Dword 0 of the completion queue entry for that command and the Deterministic Threshold Configuration data structure is returned.

Figure 346: Predictable Latency Mode Config – Command Dword 11

Bits	Description
31:16	Reserved
15:00	**NVM Set Identifier:** This field specifies the NVM Set to be modified.

Figure 347: Predictable Latency Mode Config – Command Dword 12

Bits	Description
31:01	Reserved
00	**Predictable Latency Enable:** If this bit is set to '1', then Predictable Latency Mode (refer to section 8.16) is enabled for the NVM Set specified. If this bit is cleared to '0', then Predictable Latency Mode is disabled for the NVM Set specified.

Predictable Latency Events (refer to section 5.16.1.12) are configured as described in Figure 348.

Figure 348: Predictable Latency Mode – Deterministic Threshold Configuration Data Structure

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>01:00</td><td>Enable Event: This field specifies whether an entry shall be added to the Predictable Latency Event Aggregate log page for the associated event. If a bit is set to ‘1’, then an entry shall be added if the specified event occurs. If a bit is cleared to ‘0’, then an entry shall not be added if the specified event occurs.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>00</td><td>DTWIN Reads Warning</td></tr>
<tr><td>01</td><td>DTWIN Writes Warning</td></tr>
<tr><td>02</td><td>DTWIN Time Warning</td></tr>
<tr><td>03 to 13</td><td>Reserved</td></tr>
<tr><td>14</td><td>Autonomous transition from DTWIN to NDWIN due to typical or maximum value exceeded.</td></tr>
<tr><td>15</td><td>Autonomous transition from DTWIN to NDWIN due to Deterministic Excursion.</td></tr>
</table></td></tr>
<tr><td>31:02</td><td>Reserved</td></tr>
<tr><td>39:32</td><td>DTWIN Reads Threshold: If the value of DTWIN Reads Estimate falls below this value and the DTWIN Reads Warning is enabled, then the ‘DTWIN Reads Warning’ event is set in the Predictable Latency Per NVM Set log page for the affected NVM Set.</td></tr>
<tr><td>47:40</td><td>DTWIN Writes Threshold: If the value of DTWIN Writes Estimate falls below this value and the DTWIN Writes Warning is enabled, then the ‘DTWIN Writes Warning’ event is set in the Predictable Latency Per NVM Set log page for the affected NVM Set.</td></tr>
<tr><td>55:48</td><td>DTWIN Time Threshold: If the value of DTWIN Time Estimate falls below this value and the DTWIN Time Warning is enabled, then the ‘DTWIN Time Warning’ event is set in the Predictable Latency Per NVM Set log page for the affected NVM Set.</td></tr>
<tr><td>511:56</td><td>Reserved</td></tr>
</table>

5.27.1.17 Predictable Latency Mode Window (Feature Identifier 14h)

This Feature is used to set the window for the specified NVM Set and its associated namespaces if the NVM Set is configured in Predictable Latency Mode (refer to section 8.16). The attributes are specified in Command Dword 11 and Command Dword 12. If Predictable Latency Mode is not enabled, then the controller shall abort the command with a status code of Invalid Field in Command.

The transition to the window selected is complete when the Set Features command completes successfully. A transition to the Deterministic Window may be delayed (i.e., the Set Features command completion is delayed) if the minimum time has not been spent in the Non-Deterministic Window.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 350 are returned in Dword 0 of the completion queue entry for that command. If Predictable Latency Mode is not enabled, then the controller shall abort the command with a status code of Invalid Field in Command.

Figure 349: Predictable Latency Mode Window – Command Dword 11

Bits	Description
31:16	Reserved
15:00	**NVM Set Identifier:** This field specifies the NVM Set to be modified.

Figure 350: Predictable Latency Mode Window – Command Dword 12

Bits	Description
31:03	Reserved

Figure 350: Predictable Latency Mode Window – Command Dword 12

Bits	Description
02:00	**Window Select:** This field selects or indicates the window used by all namespaces in the NVM Set. **Value** / **Definition** 000b – Reserved 001b – Deterministic Window (DTWIN) 010b – Non-Deterministic Window (NDWIN) 011b to 111b – Reserved

5.27.1.18 Host Behavior Support (Feature Identifier 16h)

This Feature enables use of controller functionality that is associated with and depends upon specific host behavior that may or may not be supported by all hosts. A controller does not use such functionality unless the host has indicated that the host supports the specific host behavior upon which the functionality depends. The host indicates that support to the controller by setting a field in this Feature. That host action enables controller use of the associated functionality with that host. A controller shall not use functionality with a host that has not indicated support for the associated specific host behavior upon which that controller functionality depends. The attributes in Figure 351 are transferred in the data buffer.

For example, the Command Interrupted status code is associated with and depends upon the specific host behavior that the host is expected to retry commands that are aborted with that status code. That command retry behavior may or may not be supported by all hosts (e.g., hosts compliant with versions 1.3 and earlier of the NVM Express Base Specification are unlikely to retry commands aborted with the Command Interrupted status code as that status code was introduced after NVM Express Base Specification, Revision 1.3). A host that supports that command retry behavior indicates its support to the controller by setting a field to 1h in the Host Behavior Support Feature. Setting that field to 1h enables controller use of the Command Interrupted status code, with the result that this status code is used only with hosts that have indicated support for the associated command retry behavior.

This Feature is not saveable (refer to Figure 195). The default value of this Feature shall be all bytes cleared to 0h.

After a successful completion of a Set Features command for this Feature, the controller may use controller-to-host functionality that depends on specific host behavior as indicated by the attributes. If multiple Set Features commands for this Feature are processed by the controller, only information from the most recent successful command is retained (i.e., subsequent commands replace information provided by previous commands).

If a Get Features command is submitted for this Feature, the attributes specified in Figure 351 are returned in the data buffer for that command.

Figure 351: Host Behavior Support – Data Structure

Bytes	Description
00	**Advanced Command Retry Enable (ACRE):** If set to 1h, then the Command Interrupted status code is enabled (refer to Figure 94) and command retry delays are enabled. The controller may use the Command Interrupted status code and may indicate a command retry delay by setting the Command Retry Delay (CRD) field to a non-zero value in the Status field of a completion queue entry, refer to Figure 92. A host that sets this field to 1h indicates host support for the command retry behaviors that are specified for both the Command Interrupted status code and non-zero values in the CRD field. If cleared to 0h, then both the Command Interrupted status code and command retry delays are disabled. The controller shall not use the Command Interrupted status code, and shall clear the CRD field to 0h in all CQEs. All values other than 0h and 1h are reserved.

Figure 351: Host Behavior Support – Data Structure

Bytes	Description
01	**Extended Telemetry Data Area 4 Supported (ETDAS):** If set to 1h, then Telemetry Host-Initiated Data Area 4 and Telemetry Controller-Initiated Data Area 4 are supported by the host. If bit 6 of the Log Page Attributes field is set to '1', then the controller may populate Telemetry Host-Initiated Data Area 4 (refer to section 5.16.1.8) and the Telemetry Controller-Initiated Data Area 4 (refer to section 5.16.1.9). If cleared to 0h, then Telemetry Host-Initiated Data Area 4 and Telemetry Controller-Initiated Data Area 4 are not supported by the host. All values other than 0h and 1h are reserved.
02	**LBA Format Extension Enable (LBAFEE):** I/O Command Set specific definition. Refer to the applicable I/O Command Set specification for details. All values other than 0h and 1h are reserved.
511:03	Reserved

5.27.1.19 Sanitize Config (Feature Identifier 17h)

This Feature controls behavior of the Sanitize command and sanitize operations. The scope of this Feature is the NVM subsystem.

The attributes are specified in Command Dword 11.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 352 are returned in Dword 0 of the completion queue entry for that command.

If this Feature is not saveable (refer to Figure 195), then the default value of the NODRM attribute shall be cleared to '0' (i.e., No-Deallocate Error Response Mode).

If the capabilities of the Sanitize Config Feature Identifier are both changeable and saveable (refer to section 4.2), then the host is able to configure this Feature when initially provisioning a device.

Figure 352: Sanitize Config – Command Dword 11

Bits	Description
31:01	Reserved
00	**No-Deallocate Response Mode (NODRM):** If the No-Deallocate Inhibited bit) is set to '1' in the Sanitize Capabilities field of the Identify Controller data structure (refer to Figure 275), then this bit defines the response of the controller to a Sanitize command processed with the No-Deallocate After Sanitize bit (refer to Figure 303) set to '1'. If this bit is set to '1' (i.e., No-Deallocate Warning Response Mode), then the controller shall process such Sanitize commands, and if the resulting sanitize operation is completed successfully, then bits 2:0 of the Sanitize Status field in the Sanitize Status log page shall be set to 100b (refer to Figure 267). If this bit is cleared to '0' (i.e., No-Deallocate Error Response Mode), then the controller shall abort such Sanitize commands with a status code of Invalid Field in Command. If the No-Deallocate Inhibited bit in the Sanitize Capabilities field of the Identify Controller data structure (refer to Figure 275) is cleared to '0', then this bit has no effect.

5.27.1.20 Endurance Group Event Configuration (Feature Identifier 18h)

This Feature controls the events that trigger adding an Endurance Group Event Aggregate Log Change Notices event to the Endurance Group Event Aggregate log. This Feature may be used to disable reporting events in the case of a persistent condition (refer to section 5.2). If the condition for an event is true when the corresponding notice is enabled, then an event is sent to the host. The attributes are specified in Command Dword 11.

If a Get Features command is submitted for this Feature, the Endurance Group Critical Warnings field in Command Dword 11 is not used and the attributes specified in Figure 353 are returned in Dword 0 of the completion queue entry for that command.

Figure 353: Asynchronous Event Configuration – Command Dword 11

Bits	Description
31:24	Reserved
23:16	**Endurance Group Critical Warnings:** This field determines whether an event entry for an Endurance Group (refer to section 3.2.3) is added to the Endurance Group Event Aggregate log page (refer to section 5.16.1.15) for the corresponding Critical Warning specified in the Endurance Group Information log page (refer to Figure 217). If a bit is set to '1', then an entry is added when the corresponding critical warning bit is set to '1' in the Endurance Group Information log page. If a bit is cleared to '0', then an entry is not added when the corresponding critical warning bit is set to '1' in the Endurance Group Information log page.
15:00	**Endurance Group Identifier (ENDGID):** This field indicates the Endurance Group for which asynchronous events are being configured. If this field is cleared to 0h, then the Endurance Group Critical Warnings field is not used.

If a bit is set to '1' in the Endurance Group Critical Warnings field which corresponds to a reserved bit in the Critical Warning field of the Endurance Group Information log page (refer to Figure 217), then the Set Features command shall be aborted with a status code of Invalid Field in Command.

If the Endurance Group Identifier specifies an Endurance Group that does not exist, then the Set Features or Get Features command shall be aborted with a status code of Invalid Field in Command.

5.27.1.21 I/O Command Set Profile (Feature Identifier 19h)

This Feature specifies the I/O Command Sets that may be used by the controller when all supported I/O Command Sets (110b) are selected in CC.CSS. This Feature shall be implemented if CAP.CSS bit 6 is set to '1'. When CC.CSS is set to any value other than 110b, then this Feature has no effect and the I/O Command Sets that may be used by the controller are specified by CC.CSS. If CC.CSS is set to any value other than 110b and the controller receives a Set Features command for this Feature, then this command has no effect and returns a status code of Successful Completion.

When all supported I/O Command Sets (110b) is selected in CC.CSS, the value of this Feature specifies the index of the I/O Command Set Combination in the Identify I/O Command Set data structure that is used. Refer to section 5.17.2.21 for more information. The Index is specified in the I/O Command Set Combination Index field of Command Dword 11 (refer to Figure 354). If any namespace attached to the controller uses an I/O Command Set that is not supported by the specified I/O Command Set combination, then the controller shall abort the command with a status code of I/O Command Set Combination Rejected. Upon successful completion of a Set Features command for this Feature, the controller transitions to using the specified I/O Command Set Combination.

Figure 354 I/O Command Set Profile – Command Dword 11

Bits	Description
31:09	Reserved
08:00	**I/O Command Set Combination Index (IOCSCI):** This field specifies the index of the I/O Command Set Combination that is to be used. This field is used for the Set Features command only and is ignored for the Get Features command for this Feature. The controller shall abort a command that specifies an index that corresponds to an I/O Command Set Combination that has a value of 0h with a status code of I/O Command Set Combination Rejected.

If a Get Features command is submitted for this Feature, then the attributes described in Figure 355 are returned in Dword 0 of the completion queue entry for that command.

Figure 355: I/O Command Set Profile – Completion Queue Entry Dword 0

Bits	Description
31:09	Reserved
08:00	**I/O Command Set Combination Index (IOCSCI):** This field returns the index of the currently selected I/O Command Set Combination.

5.27.1.22 Spinup Control (Feature Identifier 1Ah)

This Feature allows the host to configure the method for initial spinup for Endurance Groups that store data on rotational media (refer to section 8.20). The NVM subsystem is the scope for this feature.

If the NVM subsystem does not contain any Endurance Groups that store data on rotational media, then the controller shall abort the Set Features command and the Get Features command for this Feature with status code of Invalid Field in Command.

The method is specified in Command Dword 11 (refer to Figure 356).

Figure 356: Spinup Control – Command Dword 11

Bits	Description
31:01	Reserved
0	If set to '1', then the Spinup Control feature is enabled. If cleared to '0', then the Spinup Control feature is disabled. The setting is persistent.

If a Get Features command is submitted for this Feature, the attributes described in (refer to Figure 357) are returned in Dword 0 of the completion queue entry for that command.

Figure 357: Completion Queue Entry Dword 0

Bits	Description
31:01	Reserved
0	If set to '1', then the Spinup Control feature is enabled. If cleared to '0', then the Spinup Control feature is disabled.

5.27.1.23 Host Metadata (Feature Identifier 7Dh), (Feature Identifier 7Eh), (Feature Identifier 7Fh)

The Host Metadata features are the Enhanced Controller Metadata feature (Feature Identifier 7Dh), the Controller Metadata feature (Feature Identifier 7Eh), and the Namespace Metadata feature (Feature Identifier 7Fh).

If a Get Features command specifying one of the Host Metadata features with the SEL field set to 011b (i.e., Supported Capabilities) is submitted, then the Saveable bit in Dword 0 of the corresponding completion queue entry shall be cleared to '0' (i.e., refer to section 4.2), and the Changeable bit in Dword 0 of the corresponding completion queue entry shall be set to '1'.

If a Get Features command specifying one of the Host Metadata features, the controller shall perform additional actions specified in Figure 358.

Figure 358: Get Features – Command Dword 11

Bits	Description
31:01	Reserved

Figure 358: Get Features – Command Dword 11

Bits	Description
00	**Generate Default Host Metadata (GDHM):** If set to '1', then the controller shall generate a number of vendor specific strings for the Element Types of the specified Host Metadata feature value. If the generated vendor specific string's Metadata Element Descriptor does not exist for the Host Metadata Data Structure that contains the default value of the specified Host Metadata Feature value, then the controller shall create the Metadata Element Descriptor in the Host Metadata Data Structure that contains the default value with the generated vendor specific string. If the generated vendor specific string's Metadata Element Descriptor does exist for the Host Metadata Data Structure that contains the default value of the specified Host Metadata Feature value, then the controller shall replace the Metadata Element Descriptor with the generated vendor specific string. If the number of vendor specific strings generated is 0h, then the default value for the Number of Metadata Element Descriptors for the specified Host Metadata feature shall be 0h. If the number of vendor specific strings generated is not 0h, then the Host Metadata Data Structure that contains the default value for the Number of Metadata Element Descriptors of the specified Host Metadata Feature value shall be the number of vendor specific strings created. If cleared to '0', then the controller shall not generate any vendor specific strings for the Element Types of the specified Host Metadata feature.

The host issues a Set Features command specifying one of the Host Metadata features containing a Host Metadata data structure (refer to Figure 360). The host receives a Host Metadata data structure via the Get Features command. The content of the strings in the Host Metadata data structure are vendor specific.

The Action is specified in Command Dword 11 as shown in Figure 359.

Figure 359: Set Features – Command Dword 11

Bits	Description
31:15	Reserved

Figure 359: Set Features – Command Dword 11

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>14:13</td><td>Element Action (EA): This field specifies the action to perform on the specified Host Metadata Feature value for each Metadata Element Descriptor data structure contained in the Host Metadata data structure.

<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>Add/Replace Entry</td></tr>
<tr><td>01b</td><td>Delete Entry Multiple</td></tr>
<tr><td>10b</td><td>Add Entry Multiple</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table>

If the Element Action field is cleared to 00b (Add/Replace Entry) and the Metadata Element Descriptor with the specified Element Type (refer to Figure 361) does not exist in the specified Host Metadata Feature value, then the Controller shall create the descriptor in the specified Host Metadata Feature value with the value in the Host Metadata data structure.

If the Element Action field is cleared to 00b (Add/Replace Entry) and one Metadata Element Descriptor with the specified Element Type exists in the specified Host Metadata Feature value, then the Controller shall replace with the value in the specified Host Metadata data structure.

If the Element Action field is cleared to 00b (Add/Replace Entry) and the Feature Identifier field is set to Enhanced Controller Metadata, then the controller shall abort the Set Features command with a status code of Invalid Field in Command and shall not change any Host Metadata Feature value.

If the Element Action field is set to 01b (Delete Entry Multiple), then the Controller shall delete all the specified Metadata Element Descriptors from the specified Host Metadata Feature value, if any. If none of the specified Metadata Element Descriptors are present in the specified Host Metadata Feature value, then the controller shall complete the Set Features command with a status code of Successful Completion and shall not change any Host Metadata Feature value.

If the Element Action field is set to 10b (Add Entry Multiple), the Feature Identifier field is set to Enhanced Controller Metadata, and no Metadata Element Descriptor with the specified Element Type exists in the Enhanced Controller Metadata Feature value, then the controller shall create new Metadata Element Descriptors in the Enhanced Controller Metadata Feature value with the Element Type and the value specified in the Host Metadata data structure.

If the Element Action field is set to 10b (Add Entry Multiple), the Feature Identifier field is set to Enhanced Controller Metadata, and one or more Metadata Element Descriptors with the specified Element Type exists in the Enhanced Controller Metadata Feature value, then the controller shall add the specified Metadata Element to the Enhanced Controller Metadata Feature value and shall not modify any existing Metadata Element Descriptors.

If the Element Action field is set to 10b (Add Entry Multiple) and the Feature Identifier field is not set to Enhanced Controller Metadata, then the controller shall abort the Set Features command with a status code of Invalid Field in Command and shall not change the Host Metadata Feature value.</td></tr>
<tr><td>12:00</td><td>Reserved</td></tr>
</table>

Metadata Element Descriptors may be added, replaced, or deleted based on the action specified in the Element Action field. Modification of the Host Metadata Feature value shall be performed by the controller in an atomic manner.

If a Set Features command is submitted for a Host Metadata Feature, a Host Metadata data structure, defined in Figure 360, is transferred in the data buffer for the command. The Host Metadata data structure is 4 KiB in size and contains zero or more Metadata Element Descriptors. If host software attempts to add or replace a Metadata Element that causes the Host Metadata Feature value of the specified feature to grow larger than 4 KiB, then the controller shall abort the command with a status code of Invalid Field in Command.

If the host receives a Host Metadata data structure via the Get Features command, then all of the Metadata Element Descriptors present for the specified feature are added to a Host Metadata data structure (refer to Figure 360) and returned in the data buffer for that command. The data buffer size is equal to the size of the Host Metadata data structure that is 4 KiB in size.

Figure 360: Host Metadata Data Structure

Bytes	Description
00	**Number of Metadata Element Descriptors:** This field contains the number of Metadata Element descriptors in the data structure.
01	Reserved
x:02	**Metadata Element Descriptor 0:** This field contains the first Metadata Element descriptor or 0h if there are no entries.
y:x+1	**Metadata Element Descriptor 1:** This field contains the second Metadata Element descriptor or 0h if there is only 1 entry.
…	…
4095:z	**Metadata Element Descriptor N:** This field contains the (N+1)th Metadata Element descriptor or 0h if there are fewer than N+1 entries.

If the Feature Identifier field specifies Controller Metadata or Namespace Metadata, then the Host Metadata data structure may contain at most one Metadata Element Descriptor of each Element Type. If the Feature Identifier field specifies Enhanced Controller Metadata, then a Host Metadata data structure may contain more than one Metadata Element Descriptor of each Element Type. Each Metadata Element Descriptor contains the data structure shown in Figure 361.

Figure 361: Metadata Element Descriptor

Bit	Description
31 + (Element Length*8) :32	**Element Value (EVAL):** This field specifies the value for the element.
31:16	**Element Length (ELEN):** This field specifies the length of the Element Value field in bytes. This field shall be cleared to 0h when deleting an entry (i.e., the EA field is set to 01b in Command Dword 11). This field should be non-zero when adding/updating an entry (i.e., the EA field is cleared to 00b). If this field is cleared to 0h when adding/updating an entry, then the controller behavior is undefined.
15:12	Reserved
11:08	**Element Revision (ER):** This field specifies the revision of this element value. Unless specified otherwise elsewhere in this specification, all Metadata Element Descriptors shall clear this field to 0h.
07:05	Reserved
04:00	**Element Type (ET):** This field specifies the type of metadata stored in the descriptor. **Value** / **Definition** 00h: Reserved 01h to 17h: Element Types defined by this specification. Enhanced Controller Metadata Element and Controller Metadata Element types are defined in Figure 362. Namespace Metadata Element types are defined in Figure 363. 18h to 1Fh: Vendor Specific

5.27.1.23.1 Enhanced Controller Metadata (Feature Identifier 7Dh)

This feature is used to store metadata about the host platform in an NVM subsystem for later retrieval.

The metadata element types defined in Figure 362 are used by this feature.

Figure 362: Controller Metadata Element Types

Value	Definition
00h	Reserved

Figure 362: Controller Metadata Element Types

Value	Definition
01h	**Operating System Controller Name:** The name of the controller in the operating system as a UTF-8 string.
02h	**Operating System Driver Name:** The name of the driver in the operating system as a UTF-8 string.
03h	**Operating System Driver Version:** The version of the driver in the operating system as a UTF-8 string.
04h	**Pre-boot Controller Name:** The name of the controller in the pre-boot environment as a UTF-8 string.
05h	**Pre-boot Driver Name:** The name of the driver in the pre-boot environment as a UTF-8 string.
06h	**Pre-boot Driver Version:** The version of the driver in the pre-boot environment as a UTF-8 string.
07h	**System Processor Model:** The model of the processor as a UTF-8 string.
08h	**Chipset Driver Name:** The chipset driver name as a UTF-8 string.
09h	**Chipset Driver Version:** The chipset driver version as a UTF-8 string.
0Ah	**Operating System Name and Build:** The operating system name and build as a UTF-8 string.
0Bh	**System Product Name:** The system product name as a UTF-8 string.
0Ch	**Firmware Version:** The host firmware (e.g., UEFI) version as a UTF-8 string.
0Dh	**Operating System Driver Filename:** The operating system driver filename as a UTF-8 string.
0Eh	**Display Driver Name:** The display driver name as a UTF-8 string.
0Fh	**Display Driver Version:** The display driver version as a UTF-8 string.
10h	**Host-Determined Failure Record:** A failure record (e.g., the reason the host has flagged a failure for an NVMe Storage Device (refer to the NVM Express Management Interface Specification) FRU which may be used for failure analysis) as a UTF-8 string.
11h to 17h	Reserved
18h to 1Fh	Vendor Specific

Refer to section 5.27.1.23 for the definitions of Command Dword 11 and the Host Metadata Data Structure.

The default value for the Number of Metadata Element Descriptors of the Enhanced Controller Metadata Feature shall be 0h on a Controller Level Reset.

If a Get Features command with the SEL field set to 011b (i.e., Supported Capabilities) with the Enhanced Controller Metadata Feature value is submitted, then the NS Specific bit in Dword 0 of the corresponding completion queue entry shall be cleared to ‘0’.

5.27.1.23.2Controller Metadata (Feature Identifier 7Eh)

This feature is used to store metadata about the host platform in an NVM subsystem for later retrieval.

The Controller Metadata Feature provides backward compatibility with Management Controllers (refer to the NVM Express Management Interface Specification) compliant with version 1.1 and earlier versions of the NVM Express Management Interface Specification.

If a controller supports both the Enhanced Controller Metadata Feature and the Controller Metadata Feature, then the Controller Metadata Feature should not be used by the host.

The metadata element types defined in Figure 362 are used by this feature.

Refer to section 5.27.1.23 for the definitions of Command Dword 11 and the Host Metadata Data Structure.

If a Set Features command’s Element Action field of Command Dword 11 is set to 10b (Add Entry Multiple), then the controller shall abort the command with a status code of Invalid Field in Command and shall not change the Host Metadata Feature value.

The default value for the Number of Metadata Element Descriptors of the Controller Metadata Feature shall be 0h on a Controller Level Reset.

If a Get Features command with the SEL field set to 011b (i.e., Supported Capabilities) with the Controller Metadata Feature value is submitted, then the NS Specific bit in Dword 0 of the corresponding completion queue entry shall be cleared to ‘0’.

5.27.1.23.3 Namespace Metadata (Feature Identifier 7Fh)

This feature is used to store metadata about a namespace associated with a controller in the NVM subsystem for later retrieval. This feature is namespace specific. The Add Entry Multiple action is prohibited for this feature.

Figure 363: Namespace Metadata Element Types

Value	Definition
00h	Reserved
01h	**Operating System Namespace Name:** The name of the namespace in the operating system as a UTF-8 string.
02h	**Pre-boot Namespace Name:** The name of the namespace in the pre-boot environment as a UTF-8 string.
03h	**Operating System Namespace Name Qualifier 1:** The first qualifier of the Operating System Namespace Name as a UTF-8 string.
04h	**Operating System Namespace Name Qualifier 2:** The second qualifier of the Operating System Namespace Name as a UTF-8 string.
05h to 17h	Reserved
18h to 1Fh	Vendor Specific

Refer to section 5.27.1.23 for the definitions of Command Dword 11 and the Host Metadata Data Structure.

If a Get Features command with the SEL field set to 011b (i.e., Supported Capabilities) with the Namespace Metadata Feature value is submitted, then the NS Specific bit in Dword 0 of the corresponding completion queue entry shall be set to ‘1’.

5.27.1.24 Software Progress Marker (Feature Identifier 80h)

This Feature is a software progress marker. The software progress marker is persistent across power states. This information may be used to indicate to an OS software driver whether there have been issues with the OS successfully loading. The attributes are specified in Command Dword 11.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 364 are returned in Dword 0 of the completion queue entry for that command.

Figure 364: Software Progress Marker – Command Dword 11

Bits	Description
31:08	Reserved
07:00	**Pre-boot Software Load Count (PBSLC):** Indicates the load count of pre-boot software. After successfully loading and initializing the controller, pre-boot software should set this field to one more than the previous value of the Pre-boot Software Load Count. If the previous value is 255, then the value should not be updated by pre-boot software (i.e., the value does not wrap to 0). OS driver software should set this field to 0h after the OS has successfully been initialized.

5.27.1.25 Host Identifier (Feature Identifier 81h)

This feature allows the host to register a Host Identifier with the controller. The Host Identifier is used by the controller to determine whether other controllers in the NVM subsystem are associated with the same host. The Host Identifier may be used to designate host elements that access an NVM subsystem independently of each other or for reservations.

The Host Identifier is contained in the data structure indicated in Figure 366. The attributes are specified in Command Dword 11. If a Get Features command is issued for this Feature, the data structure specified in Figure 366 is returned in the data buffer for that command.

The requirements and use of the Host Identifier feature is dependent on whether the NVMe over PCIe implementation or NVMe over Fabrics implementation are supported. Refer to section 5.27.1.25.1 and section 5.27.1.25.2.

Figure 365: Host Identifier – Command Dword 11

Bits	Description
31:01	Reserved
00	**Enable Extended Host Identifier (EXHID):** If set to '1', then the host is requesting the use of an extended 128-bit Host Identifier. If cleared to '0', then the host is requesting the use of a 64-bit Host Identifier. NVMe over Fabrics implementations shall use an extended 128-bit Host Identifier. If the controller does not support a 128-bit Host Identifier as indicated in the Controller Attributes field in the Identify Controller data structure and the host sets this bit to '1', then a status code of Invalid Field in Command shall be returned. If the controller does not support a 64-bit Host Identifier (e.g., the device is an NVMe over Fabrics device) and the host clears this bit to '0', then a status code of Invalid Field in Command shall be returned. If the NVM subsystem supports a 64-bit Host Identifier, supports a 128-bit Host Identifier and detects that another controller in the NVM subsystem is already using a non-zero Host Identifier of a different size than the size requested in this command, then a status code of Host Identifier Inconsistent Format shall be returned.

Figure 366: Host Identifier – Data Structure Entry

Bytes	Description
15:00	**Host Identifier (HOSTID):** This field specifies a 64-bit or 128-bit identifier that uniquely identifies the host associated with the controller within the NVM subsystem. The host provides an 8 byte or 16 byte data structure depending on the value specified in the Enable Extended Host Identifier bit. The value of the Host Identifier used by a host, the method used to select this value, and the method used to ensure uniqueness are outside the scope of this specification. Controllers in an NVM subsystem that have the same Host Identifier are assumed to be associated with the same host and have the same reservation and registration rights. A Host Identifier value of 0h indicates that the host is not associated with any other controller in the NVM subsystem.

5.27.1.25.1PCIe Transport Implementations

The Host Identifier is an optional feature when implemented on a controller using a PCIe transport. The controller may support a 64-bit Host Identifier and/or an extended 128-bit Host Identifier. It is recommended that implementations support the extended 128-bit Host Identifier as indicated in the Controller Attributes field in the Identify Controller data structure. The Host Identifier may be modified at any time using a Set Features command causing the controller to be logically remapped from the original host associated with the old Host Identifier to a new host associated with the new Host Identifier.

A Host Identifier value of 0h is a valid value that indicates that the host associated with the controller is not associated with any other controller in the NVM subsystem. Specifically, two controllers in an NVM subsystem that both have a Host Identifier of 0h indicates that the controllers are associated with different hosts. Using a Host Identifier value of 0h is a valid configuration for the reservations feature. However, reservations and registrations associated with a Host Identifier of 0h do not persist across a Controller Level Reset since a host that uses a Host Identifier of 0h is treated as a different host after a Controller Level Reset.

5.27.1.25.2NVMe over Fabrics Implementations

The Host Identifier is a mandatory feature in NVMe over Fabrics implementations. The Host Identifier shall be an extended 128-bit Host Identifier. The Host Identifier shall be set to a non-zero value in the Fabrics Connect command. The Host Identifier shall not be modified. A Set Features command specifying the Host Identifier Feature shall be aborted with a status code of Command Sequence Error. A Get Features command specifying the Host Identifier Feature shall return the value set in the Fabrics Connect command.

5.27.1.26 Reservation Notification Mask (Feature Identifier 82h)

This Feature controls the masking of reservation notifications on a per namespace basis. A Reservation Notification log page is created whenever a reservation notification occurs on a namespace and the corresponding reservation notification type is not masked on that namespace by this Feature. If reservations are supported by the controller, then this Feature shall be supported. The attributes are specified in Command Dword 11.

A Set Features command that uses a namespace ID other than FFFFFFFFh modifies the reservation notification mask for the corresponding namespace only. A Set Features command that uses a namespace ID of FFFFFFFFh modifies the reservation notification mask of all namespaces that are attached to the controller and that support reservations. A Get Features command that uses a namespace ID other than FFFFFFFFh returns the reservation notification mask for the corresponding namespace. A Get Features command that uses a namespace ID of FFFFFFFFh should be aborted with status code of Invalid Field in Command. If a Set Features command or a Get Features command attempts to access the Reservation Notification Mask on a namespace that does not support reservations or is invalid, then that command is aborted with status code of Invalid Field in Command.

If a Get Features command successfully completes for this Feature, the attributes specified in Figure 367 are returned in Dword 0 of the completion queue entry for that command.

Figure 367: Reservation Notification Configuration – Command Dword 11

Bits	Description
31:04	Reserved
03	**Mask Reservation Preempted Notification (RESPRE):** If set to ‘1’, then mask the reporting of reservation preempted notification by the controller. If cleared to ‘0’, then the notification is not masked and a Reservation Notification log page is created whenever notification occurs.
02	**Mask Reservation Released Notification (RESREL):** If set to ‘1’, then mask the reporting of reservation released notification by the controller. If cleared to ‘0’, then the notification is not masked and a Reservation Notification log page is created whenever the notification occurs.
01	**Mask Registration Preempted Notification (REGPRE):** If set to ‘1’, then mask the reporting of registration preempted notification by the controller. If cleared to ‘0’, then the notification is not masked and a Reservation Notification log page is created whenever the notification occurs.
00	Reserved

5.27.1.27 Reservation Persistence (Feature Identifier 83h)

Each namespace that supports reservations has a Persist Through Power Loss (PTPL) state that may be modified using either a Set Features command or a Reservation Register command (refer to section 7.3). The Reservation Persistence feature attributes are specified in Command Dword 11.

The PTPL state is contained in the Reservation Persistence Feature that is namespace specific. A Set Features command that uses the namespace ID FFFFFFFFh modifies the PTPL state associated with all namespaces that are attached to the controller and that support PTPL (i.e., support reservations). A Set Features command that uses a valid namespace ID other than FFFFFFFFh and corresponds to a namespace that supports reservations, modifies the PTPL state for that namespace. A Get Features command that uses a namespace ID of FFFFFFFFh should be aborted with a status code of Invalid Field in Command. A Get Features command that uses a valid namespace ID other than FFFFFFFFh and corresponds to a namespace that supports PTPL, returns the PTPL state for that namespace. If a Set Features command or a Get Features command using a namespace ID other than FFFFFFFFh attempts to access the PTPL state for a namespace that does not support this Feature Identifier, then the command is aborted with status code of Invalid Field in Command.

This Feature should not be saveable (refer to Figure 195). If this Feature is saveable, then the host should set the current value and the saved value to the same value.

If a Get Features command successfully completes for this Feature Identifier, the attributes specified in Figure 368 are returned in Dword 0 of the completion queue entry for that command

Figure 368: Reservation Persistence Configuration – Command Dword 11

Bits	Description
31:01	Reserved
00	**Persist Through Power Loss (PTPL):** If set to '1', then reservations and registrants persist across a power loss. If cleared to '0', then reservations are released and registrants are cleared on a power loss.

5.27.1.28 Namespace Write Protection Config (Feature Identifier 84h)

This Feature is used by the host to configure the namespace write protection state or to determine the write protection state of a namespace. Refer to section 8.12 for definition and behaviors of the namespace write protection states. The settings are specified in Command Dword 11.

This Feature is not saveable (refer to Figure 195). There is no default value for this Feature; the value of the Feature after a power cycle or a Controller Level Reset is determined by the write protection state of the namespace prior to the power cycle or Controller Level Reset, except for the Write Protect Until Power Cycle write protection state (refer to section 8.12).

If a Get Features command is submitted for this Feature, the attributes specified in Figure 369 are returned in Dword 0 of the completion queue entry for that command.

Figure 369: Write Protection – Command Dword 11

Bits	Description
31:03	Reserved
02:00	**Write Protection State:** This field specifies the write protection state of the specified namespace. Value / Definition: 000b – No Write Protect 001b – Write Protect 010b – Write Protect Until Power Cycle 011b – Permanent Write Protect 100b to 111b – Reserved

If a Set Features command attempts to change the namespace write protection state of a namespace that is in the Write Protect Until Power Cycle state or the Permanent Write Protect state, then the command shall abort with a status code of Feature Not Changeable.

If a Set Features command attempts to change the namespace write protection state of a namespace to the Write Protect Until Power Cycle state and bit 0 of the of the Write Protection Authentication Control field is cleared to '0', then the command shall abort with a status code of Feature Not Changeable.

If a Set Features command changes the namespace to a write protected state, then the controller shall commit all volatile write cache data and metadata associated with the specified namespace to non-volatile media as part of transitioning to the write protected state.

5.27.2 Command Completion

Upon completion of the Set Features command, the controller posts a completion queue entry to the Admin Completion Queue. If a status code of Successful Completion is returned, the completion queue entry shall not be posted until the controller has completed setting attributes associated with the Feature. Set Features command specific status values are defined in Figure 370.

Figure 370: Set Features – Command Specific Status Values

Value	Description
0Dh	**Feature Identifier Not Saveable:** The Feature Identifier specified does not support a saveable value.
0Eh	**Feature Not Changeable:** The Feature Identifier specified does not support a changeable value.

Figure 370: Set Features – Command Specific Status Values

Value	Description
0Fh	**Feature Not Namespace Specific:** The Feature Identifier specified is not namespace specific. The Feature Identifier settings apply across all namespaces.
14h	**Overlapping Range:** Command Set specific definition. Refer to each I/O Command Set specification for applicability and details.
15h	**I/O Command Set Combination Rejected:** This error indicates that the controller did not accept the request to select the requested I/O Command Set Combination.

5.28 Virtualization Management command

The Virtualization Management command is supported by primary controllers that support the Virtualization Enhancements capability. This command is used for several functions:

- Modifying Flexible Resource allocation for the primary controller;
- Assigning Flexible Resources for secondary controllers; and
- Setting the Online and Offline state for secondary controllers.

Refer to section 8.26 for more on the Virtualization Enhancements capability and the Virtualization Management command.

The Virtualization Management command uses the Command Dword 10 and Command Dword 11 fields. All other command specific fields are reserved.

If the action requested specifies a range of controller resources that:

a) does not exist;
b) is a Private Resource (e.g., VQ resources are requested when VQ resources are not supported, VI resources are requested when VI resources are not supported); or
c) is currently in use (e.g., the number of Controller Resources (NR) is greater than the number of remaining available flexible resources),

then the command is aborted with a status code of Invalid Resource Identifier.

Figure 371: Virtualization Management – Command Dword 10

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:16</td><td>Controller Identifier (CNTLID): This field indicates the controller for which controller resources are to be modified.</td></tr>
<tr><td>15:11</td><td>Reserved</td></tr>
<tr><td>10:08</td><td>Resource Type (RT): This field indicates the type of controller resource to be modified.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>000b</td><td>VQ Resources</td></tr>
<tr><td>001b</td><td>VI Resources</td></tr>
<tr><td>010b to 111b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>07:04</td><td>Reserved</td></tr>
</table>

Figure 371: Virtualization Management – Command Dword 10

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>03:00</td><td>Action (ACT): This field indicates the operation for the command to perform as described below.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0h</td><td>Reserved</td></tr>
<tr><td>1h</td><td>Primary Controller Flexible Allocation: Set the number of Flexible Resources allocated to this primary controller following the next Controller Level Reset other than a Controller Reset (i.e., CC.EN transitions from ‘1’ to ‘0’). If the Controller Identifier field does not correspond to this primary controller, then a status code of Invalid Controller Identifier is returned. This value is persistent across power cycles and resets.</td></tr>
<tr><td>2h to 6h</td><td>Reserved</td></tr>
<tr><td>7h</td><td>Secondary Controller Offline: Place the secondary controller in the Offline state and remove all Flexible Resources. If the Controller Identifier field does not correspond to a secondary controller associated with this primary controller, then a status code of Invalid Controller Identifier is returned.</td></tr>
<tr><td>8h</td><td>Secondary Controller Assign: Assign the number of controller resources specified in Number of Controller Resources to the secondary controller. If the Controller Identifier field does not correspond to a secondary controller associated with this primary controller, then an error of Invalid Controller Identifier is returned. If the secondary controller is not in the Offline state, then a status code of Invalid Secondary Controller State is returned.</td></tr>
<tr><td>9h</td><td>Secondary Controller Online: Place the secondary controller in the Online state. If the Controller Identifier field does not correspond to a secondary controller associated with this primary controller, then an error of Invalid Controller Identifier is returned. If the secondary controller is not configured appropriately (refer to section 8.26) or the primary controller is not enabled, then a status code of Invalid Secondary Controller State is returned.</td></tr>
<tr><td>Ah to Fh</td><td>Reserved</td></tr>
</table></td></tr>
</table>

Figure 372: Virtualization Management – Command Dword 11

Bits	Description
31:16	Reserved
15:00	**Number of Controller Resources (NR):** This field indicates a number of controller resources to allocate or assign.

5.28.1 Command Completion

Command specific status values associated with the Virtualization management command are defined in Figure 373.

Figure 373: Virtualization Management – Command Specific Status Values

Value	Description
1Fh	**Invalid Controller Identifier:** An invalid Controller Identifier was specified.
20h	**Invalid Secondary Controller State:** The action requested for the secondary controller is invalid based on the current state of the secondary controller and its primary controller.
21h	**Invalid Number of Controller Resources:** The specified number of Flexible Resources is invalid (e.g., the Number of Controller Resources (NR) is greater than VQ Resources Flexible Total (VQFRT) (refer to Figure 281), the Number of Controller Resources (NR) is greater than VQ Resources Flexible Secondary Maximum (VQFRSM) (refer to Figure 281)).
22h	**Invalid Resource Identifier:** At least one of the specified resource identifiers was invalid (e.g., the Number of Controller Resources (NR) is greater than the number of remaining available flexible resources).

Dword 0 of the completion queue entry contains information about the controller resources that were modified as part of the Primary Controller Flexible Allocation and Secondary Controller Assign actions. Dword 0 of the completion queue entry is defined in Figure 374.

Figure 374: Virtualization Management – Completion Queue Entry Dword 0

Bits	Description
31:16	Reserved
15:00	**Number of Controller Resources Modified (NRM):** This field indicates the number of controller resources that were allocated or assigned. The value may be smaller or larger than the number requested.

6 Fabrics Command Set

Fabrics commands are used to create queues and initialize a controller. Fabrics commands have an Opcode field of 7Fh. Fabrics commands are processed regardless of the state of controller enable (CC.EN). The Fabrics command capsule is defined in section 3.3.2.1.1 and the Fabrics response capsule and status is defined in section 3.3.2.1.2.

Restrictions on processing commands listed in Figure 375 are defined in the Admin Command Set in section 5 (e.g., while the NVM subsystem is performing a sanitize operation or processing of a Format NVM command).

Figure 375: Fabrics Command Types

Command Type by Field			Combined Command Type[2]	O/M[1]	I/O Queue[3]	Command
(07)	(06:02)	(01:00)				
Generic Command	Function	Data Transfer[4]				
0b	000 00b	00b	00h	M	No	Property Set
0b	000 00b	01b	01h	M	Yes	Connect[5]
0b	000 01b	00b	04h	M	No	Property Get
0b	000 01b	01b	05h	O	Yes	Authentication Send
0b	000 01b	10b	06h	O	Yes	Authentication Receive
0b	000 10b	00b	08h	O	Yes	Disconnect
	Vendor Specific					
1b	na	na	C0h to FFh	O		Vendor specific

Notes:
1. O/M definition: O = Optional, M = Mandatory.
2. Opcodes not listed are reserved.
3. All Fabrics commands, other than the Disconnect command, may be submitted on the Admin Queue. The I/O Queue supports Fabrics commands as specified in this column. If a Fabrics command that is not supported on an I/O Queue is sent on an I/O Queue, that command shall be aborted with a status code of Invalid Field in Command.
4. 00b = no data transfer; 01b = host to controller; 10b = controller to host; 11b = reserved
5. The Connect command is submitted and completed on the same queue that the Connect command creates. Refer to section 3.3.2.2.

6.1 Authentication Receive Command and Response

The Authentication Receive command transfers the status and data result of one or more Authentication Send commands that were previously submitted to the controller.

The association between an Authentication Receive command and previous Authentication Send commands is dependent on the Security Protocol. The format of the data to be transferred is dependent on the Security Protocol. Refer to SPC-5 for Security Protocol details.

Authentication Receive commands return the appropriate data corresponding to an Authentication Send command as defined by the rules of the Security Protocol. The Authentication Receive command data shall not be retained if there is a loss of communication between the controller and host, or if a Controller Level Reset occurs.

Figure 376 Authentication Receive Command – Submission Queue Entry

Bytes	Description
00	**Opcode (OPC):** Set to 7Fh to indicate a Fabrics command.
01	Reserved
03:02	**Command Identifier (CID):** This field specifies a unique identifier for the command. Refer to the definition in Figure 80.
04	**Fabrics Command Type (FCTYPE):** Set to 06h to indicate an Authentication Receive command.

Figure 376 Authentication Receive Command – Submission Queue Entry

Bytes	Description
23:05	Reserved
39:24	**SGL Descriptor 1 (SGL1):** This field contains a Transport SGL Data Block descriptor or a Keyed SGL Data Block descriptor that describes the entire data transfer. Refer to section 4.1.2 for the definition of SGL descriptors.
40	Reserved
41	**SP Specific 0 (SPSP0):** The value of this field contains bits 07:00 of the Security Protocol Specific field as defined in SPC-5.
42	**SP Specific 1 (SPSP1):** The value of this field contains bits 15:08 of the Security Protocol Specific field as defined in SPC-5.
43	**Security Protocol (SECP):** This field specifies the security protocol as defined in SPC-5. The controller shall abort the command with Invalid Parameter indicated if a reserved value of the Security Protocol is specified.
47:44	**Allocation Length (AL):** The value of this field is specific to the Security Protocol as defined in SPC-5 where INC_512 is cleared to '0'.
63:48	Reserved

Figure 377: Authentication Receive Response

Bytes	Description
07:00	Reserved
09:08	**SQ Head Pointer (SQHD):** Indicates the current Submission Queue Head pointer for the associated Submission Queue.
11:10	Reserved
13:12	**Command Identifier (CID):** Indicates the identifier of the command that is being completed.
15:14	**Status (STS):** Indicates status for the command. Bits: 15:01 – Status field for the command. Refer to section 3.3.3.2.1 Bits: 00 – Reserved

6.2 Authentication Send Command and Response

The Authentication Send command is used to transfer security protocol data to the controller. The data structure transferred as part of this command contains security protocol specific commands to be performed by the controller. The data structure may contain data or parameters associated with the security protocol specific commands. Status and data that is to be returned to the host for the security protocol specific commands submitted by an Authentication Send command are retrieved with the Authentication Receive command defined in section 6.1.

The association between an Authentication Send command and subsequent Authentication Receive commands is Security Protocol field dependent as defined in SPC-5.

Figure 378: Authentication Send Command – Submission Queue Entry

Bytes	Description
00	**Opcode (OPC):** Set to 7Fh to indicate a Fabrics command.
01	Reserved
03:02	**Command Identifier (CID):** This field specifies a unique identifier for the command. Refer to the definition in Figure 80.
04	**Fabrics Command Type (FCTYPE):** Set to 05h to indicate an Authentication Send command.
23:05	Reserved
39:24	**SGL Descriptor 1 (SGL1):** This field contains a Transport SGL Data Block descriptor or a Keyed SGL Data Block descriptor that describes the entire data transfer. Refer to section 4.1.2 for the definition of SGL descriptors.

Figure 378: Authentication Send Command – Submission Queue Entry

Bytes	Description
40	Reserved
41	**SP Specific 0 (SPSP0):** The value of this field contains bits 07:00 of the Security Protocol Specific field as defined in SPC-5.
42	**SP Specific 1 (SPSP1):** The value of this field contains bits 15:08 of the Security Protocol Specific field as defined in SPC-5.
43	**Security Protocol (SECP):** This field specifies the security protocol as defined in SPC-5. The controller shall abort the command with a status code of Invalid Parameter indicated if a reserved value of the Security Protocol is specified.
47:44	**Transfer Length (TL):** The value of this field is specific to the Security Protocol as defined in SPC-5 where INC_512 is cleared to '0'.
63:48	Reserved

Figure 379: Authentication Send Response

Bytes	Description
07:00	Reserved
09:08	**SQ Head Pointer (SQHD):** Indicates the current Submission Queue Head pointer for the associated Submission Queue.
11:10	Reserved
13:12	**Command Identifier (CID):** Indicates the identifier of the command that is being completed.
15:14	**Status (STS):** Indicates status for the command. Bits: 15:01 – Status field for the command. Refer to section 3.3.3.2.1 Bits: 00 – Reserved

6.3 Connect Command and Response

The Connect command is used to create a Submission and Completion Queue pair. If the Admin Queue is specified, then the Connect command establishes an association between a host and a controller. The fields for the submission queue entry are defined in Figure 380 and the fields for the data portion are defined in Figure 381.

A host that uses a single Host NQN may employ multiple Host Identifiers to designate elements of the host that access an NVM subsystem independently of each other (e.g., physical or logical partitions of the host). Alternatively, a host may employ multiple Host NQN values to cause each element to be treated as a separate host by an NVM subsystem.

If an NVM subsystem supports DH-HMAC-CHAP authentication (refer to section 6), then the Host NQN and the NVM Subsystem NQN parameters in a Connect command are required to be different. If the Host NQN and the NVM Subsystem NQN parameters in a Connect command are identical and the NVM subsystem supports DH-HMAC-CHAP authentication, then the controller shall abort the command with a status code of Connect Invalid Host.

The NVM subsystem shall not allocate a Controller ID in the range FFF0h to FFFFh as a valid Controller ID on completion of a Connect command. If the host is not allowed to establish an association to any controller in the NVM subsystem, then the controller shall abort the command with a status code of Connect Invalid Host.

If the NVM subsystem supports the dynamic controller model, then:

- the Controller ID of FFFFh is specified as the Controller ID in a Connect command for the Admin Queue. If the controller ID is not set to FFFFh, then the controller shall abort the command with a status code of Connect Invalid Parameters;
- the NVM subsystem shall allocate any available controller to the host; and
- return that allocated Controller ID in the Connect response.

If the NVM subsystem supports the static controller model, then:

- The host is able to request a specific controller in a Connect command for the Admin Queue. If the host is not allowed to establish an association to the specified controller, then the controller shall abort the command with a status code of Connect Invalid Host;
- The Controller ID of FFFEh on the Admin Queue specifies that any Controller ID may be allocated and returned in the Connect response; and
- If the host specifies a Controller ID value of FFFFh for the Admin Queue, then the controller shall abort the command with a status code of Connect Invalid Parameters.

The NVM subsystem may allocate specific controllers to particular hosts. If a host requests a controller that is not allocated to that host, then the controller shall abort the command with a status code of Connect Invalid Host. The mechanism for allocating specific controllers to particular hosts is outside the scope of this specification.

The host shall establish an association with a controller and enable the controller before establishing a connection with an I/O Queue of the controller. If the host sends a Connect command specifying a Queue ID for an Admin Queue or I/O Queue that has already been created, then the controller shall abort the command with a status code of Command Sequence Error.

The controller shall abort a Connect command with a status code of Connect Invalid Parameters if:

- the host sends a Connect command to create an I/O Queue while the controller is disabled;
- the Host Identifier, Host NQN, NVM Subsystem NQN, and the Controller ID values specified for an I/O Queue are not the same as the values specified for the associated Admin Queue in which the association between the host and controller was established;
- the Host NQN or NVM Subsystem NQN values do not match the values that the NVM subsystem is configured to support;
- there is a syntax error in the Host NQN or NVM Subsystem NQN value (refer to section 4.4); or
- the Host Identifier is cleared to 0h.

If the NVMe Subsystem Port, NVMe Transport Type or NVMe Transport Address used by the NVMe Transport (refer to section 6.3) are not the same as the values used for the associated Admin Queue in which the association between the host and controller was established, then it is possible that the Connect command is not received by an NVM subsystem. If the Connect command is received by an NVM subsystem, then:

- the NVM subsystem that receives the command may not be the same NVM subsystem to which the association between the host and controller was established (i.e., the NVMe Transport Type and NVMe Transport Address are unique to an NVM Subsystem Port); and
- the values of the NVM Subsystem NQN or Controller ID may not be valid at that NVM Subsystem Port (e.g., the NVM Subsystem NQN may specify a different NVM subsystem than the one that received that Connect command, or the Controller ID may specify a controller that is already bound to a different NVM Subsystem Port).

If this situation occurs and the Connect command is aborted, then the status code shall be set to Connect Invalid Parameters. There is no requirement that such a Connect command be received by an NVM subsystem (e.g., if the NVMe Transport Address is not a valid transport address, or is the address of a fabric endpoint that does not support NVMe over Fabrics, then the resulting error, if any, is specific to the fabric).

Submission Queue (SQ) flow control based on the SQ Head Pointer (SQHD) field in Fabrics response capsules (refer to section 3.3.2.1.2) shall be supported by all hosts and controllers. Use of SQ flow control is negotiated by the Connect command and response. A host requests that SQ flow control be disabled by setting bit 2 of the Connect Attributes field to ‘1’ in a Connect command. A controller that agrees to disable SQ flow control shall set the SQHD field to FFFFh in the response to that Connect command. A controller that does not agree to disable SQ flow control shall set the SQHD field to a value other than FFFFh in the response to that Connect command.

If the Connect command did not request that SQ flow control be disabled, then the controller shall not set the SQHD field to FFFFh in the response to that Connect command.

SQ flow control is disabled and shall not be used for a created queue pair only if:

a) bit 2 is set to '1' in the Connect Attributes field of the Connect command that creates the queue pair; and
b) the SQHD field is set to FFFFh in the response to that Connect command.

If SQ flow control is disabled, then the SQHD field is reserved in Fabrics response capsules for all command completions on that queue pair after the response that completes the Connect command.

SQ flow control is enabled and shall be used for a created queue pair if:

a) bit 2 is cleared to '0' in the Connect Attributes field of the Connect command that creates the queue pair; or
b) the SQHD field is not set to FFFFh in the response to that Connect command.

If SQ flow control is enabled, then the controller shall use the SQHD field in Fabrics response capsules for all command completions on that queue pair, except for command completions that omit the SQHD value due to use of the SQHD pointer update optimization described in section 3.3.2.7.

Figure 380: Connect Command – Submission Queue Entry

Bytes	Description
00	**Opcode (OPC):** Set to 7Fh to indicate a Fabrics command.
01	Reserved
03:02	**Command Identifier (CID):** This field specifies a unique identifier for the command. Refer to the definition in Figure 80.
04	**Fabrics Command Type (FCTYPE):** Set to 01h to indicate a Connect command.
23:05	Reserved
39:24	**SGL Descriptor 1 (SGL1):** This field contains a Transport SGL Data Block descriptor or a Keyed SGL Data Block descriptor that describes the entire data transfer. Refer to section 4.1.2 for the definition of SGL descriptors.
41:40	**Record Format (RECFMT):** Specifies the format of the Connect command capsule. The format of the record specified in this definition shall be 0h. If the NVM subsystem does not support the value specified, then a status code of Incompatible Format shall be returned.
43:42	**Queue ID (QID):** Specifies the Queue Identifier for the Admin Queue or I/O Queue to be created. The identifier is used for both the Submission and Completion Queue. The identifier for the Admin Submission Queue and Completion Queue is 0h. The identifier for an I/O Submission and Completion Queue is in the range 1 to 65,534. If the value in this field specifies the Queue ID of a queue that already exists, then the controller shall abort the command with a status code of Invalid Queue Identifier.
45:44	**Submission Queue Size (SQSIZE):** This field indicates the size of the Submission Queue to be created. If the size is 0h or larger than the controller supports, then a status code of Connect Invalid Parameters shall be returned. The maximum size of the Admin Submission Queue is specified in the Discovery Log Page Entry for the NVM subsystem. Refer to Figure 264. This is a 0's based value.

Figure 380: Connect Command – Submission Queue Entry

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>46</td><td>Connect Attributes (CATTR): This field indicates attributes for the connection.

Bits 7:4 are reserved.

Bit 3 indicates support for deleting individual I/O Queues. If this bit is set to ‘1’, then the host supports the deletion of individual I/O Queues. If this bit is cleared to ‘0’, then the host does not support the deletion of individual I/O Queues.

Bit 2 if set to ‘1’, then the host is requesting that SQ flow control be disabled. If cleared to ‘0’, then SQ flow control shall not be disabled.

Bits 1:0 indicate the priority class to use for commands within this Submission Queue. This field is only used when the weighted round robin with urgent priority class is the arbitration mechanism selected (refer to CC.AMS in Figure 46), the field is ignored if weighted round robin with urgent priority class is not used. Refer to section 3.4.4. This field is only valid for I/O Queues and shall be cleared to 00b for Admin Queue connections.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>Urgent</td></tr>
<tr><td>01b</td><td>High</td></tr>
<tr><td>10b</td><td>Medium</td></tr>
<tr><td>11b</td><td>Low</td></tr>
</table></td></tr>
<tr><td>47</td><td>Reserved</td></tr>
<tr><td>51:48</td><td>Keep Alive Timeout (KATO): In the Connect command for the Admin Queue, this field has the same definition as the Keep Alive Timeout (Keep Alive Timer) defined in section 5.27.1.12. Upon successful completion of the Connect command the controller shall enable and activate the Keep Alive timer as described in section 3.9.

In the Connect command for an I/O Queue, this field is reserved.</td></tr>
<tr><td>63:52</td><td>Reserved</td></tr>
</table>

Figure 381: Connect Command – Data

Bytes	Description
15:00	**Host Identifier (HOSTID):** This field has the same definition as the Host Identifier defined in section 5.27.1.25. The controller shall set the Host Identifer Feature to this value.
17:16	**Controller ID (CNTLID):** Specifies the controller ID requested. This field corresponds to the Controller ID (CNTLID) value returned in the Identify Controller data structure for a particular controller. If the NVM subsystem uses the dynamic controller model, then the value shall be FFFFh for the Admin Queue and any available controller may be returned. If the NVM subsystem uses the static controller model and the value is FFFEh for the Admin Queue, then any available controller may be returned.
255:18	Reserved
511:256	**NVM Subsystem NVMe Qualified Name (SUBNQN):** NVMe Qualified Name (NQN) that uniquely identifies the NVM subsystem. Refer to section 4.4.
767:512	**Host NVMe Qualified Name (HOSTNQN):** NVMe Qualified Name (NQN) that uniquely identifies the host. Refer to section 4.4.
1023:768	Reserved

The Connect response provides status for the Connect command. If a connection is established, then the Controller ID allocated to the host is returned. The Connect response is defined in Figure 382.

For a Connect command that fails, the controller shall not:

- return a status code of Invalid Field in Command; and
- add an entry to the Error Information log page.

Figure 382: Connect Response

Bytes	Description
03:00	**Status Code Specific:** The value is dependent on the status returned. Refer to Figure 383.
07:04	Reserved
09:08	**SQ Head Pointer (SQHD):** If the Connect command requested that SQ flow control be disabled, then a value of FFFFh in this field indicates that SQ flow control is disabled for the created queue pair. Otherwise, this field indicates the current Submission Queue Head pointer for the associated Submission Queue and also indicates that SQ flow control is enabled for the created queue pair.
11:10	Reserved
13:12	**Command Identifier (CID):** Indicates the identifier of the command that is being completed.
15:14	**Status (STS):** Indicates status for the command. **Bits** / **Definition** 15:01 – Status field for the command. Refer to section 3.3.3.2.1. Refer to Figure 97 for values specific to the Connect command. 00 – Reserved

Figure 383: Connect Response – Dword 0 Value Based on Status Code

Status Code	Definition of Dword 0
Successful Completion	**Bytes** / **Description** 01:00 – **Controller ID (CNTLID):** Specifies the controller ID allocated to the host. If a particular controller was specified in the CNTLID field of the Connect command, then this field shall contain the same value. 03:02 – **Authentication and Security Requirements (AUTHREQ):** Specifies the NVMe in-band authentication and security requirements. The field is bit significant. If all bits are cleared to '0', then no requirements are specified. **Bits** / **Definition** 15:03 – Reserved 02 – **Authentication and Secure Channel Required (ASCR):** If this bit is set to '1', then authentication using NVMe over Fabrics Authentication protocols followed by secure channel establishment is required and the ATR bit should be cleared to '0'. If this bit is cleared to '0', then authentication using NVMe over Fabrics Authentication protocols followed by secure channel establishment is not required. 01 – **Authentication Transaction Required (ATR):** If this bit is set to '1', then authentication using NVMe over Fabrics Authentication protocols is required. If this bit is cleared to '0', then authentication using NVMe over Fabrics Authentication protocols is not required. 00 – Obsolete.
Connect Invalid Parameters	**Bytes** / **Description** 01:00 – **Invalid Parameter Offset (IPO):** If an invalid parameter is reported, then this field specifies the offset in bytes to the invalid parameter from the start of the SQE or the data. 02 – **Invalid Attributes (IATTR):** Specifies attributes of the invalid field parameter. Bits 7:1 are reserved. Bit 0 if cleared to '0', then the invalid parameter is specified from the start of the SQE. Bit 0 if set to '1', then the invalid parameter is specified from the start of the data. 03 – Reserved
All Other Status Values	**Bytes** / **Description** 03:00 – Reserved

6.4 Disconnect Command and Response

The Disconnect command is used to delete the I/O Queue on which the command is submitted. If a Disconnect command is submitted on an Admin Queue, then the controller shall abort the command with a status code of Invalid Queue Type. If the controller is not able to delete the I/O Queue, then the controller shall abort the command with a status code of Controller Busy. The fields for the submission queue entry are defined in Figure 384.

The NVMe Transport connection is not deleted upon issuance of a Disconnect command; the host and controller may delete the NVMe Transport connection and associated resources after all NVMe Queues (I/O Queues and/or Admin Queue) associated with that NVMe Transport connection have been deleted (refer to section 3.3.2.4).

The completion queue entry for the Disconnect command shall be the last entry submitted to the I/O Queue Completion queue by the controller (i.e., no completion queue entries shall be submitted to the I/O Queue Completion Queue after the completion queue entry for the Disconnect command). The controller shall not perform command processing for any command on an I/O queue after sending the completion queue entry for the Disconnect command.

The host should not submit commands to an I/O Submission Queue after the submission of a Disconnect command to that I/O Submission Queue; submitting commands to an I/O Queue after a Disconnect command is submitted to that I/O Queue results in undefined behavior.

Figure 384: Disconnect Command and Response

Bytes	Description
00	**Opcode (OPC):** Set to 7Fh to indicate a Fabrics command.
01	Reserved
03:02	**Command Identifier (CID):** This field specifies a unique identifier for the command. Refer to the definition in Figure 80.
04	**Fabrics Command Type (FCTYPE):** Set to 08h to indicate a Disconnect command.
23:05	Reserved
39:24	**SGL Descriptor 1 (SGL1):** This field is reserved, as there is no data transferred by this command.
41:40	**Record Format (RECFMT):** Specifies the format of the Disconnect command capsule. The format of the record specified in this definition shall be 0h. If the NVM subsystem does not support the value specified, then a status code of Incompatible Format shall be returned.
63:48	Reserved

The Disconnect response provides status for the Disconnect command. The Disconnect response is defined in Figure 385.

Figure 385: Disconnect Response

Bytes	Description
07:00	Reserved
09:08	**SQ Head Pointer (SQHD):** Indicates the current Submission Queue Head pointer for the associated Submission Queue.
11:10	Reserved
13:12	**Command Identifier (CID):** Indicates the identifier of the command that is being completed.
15:14	**Status (STS):** Indicates status for the command. **Bits** / **Definition** 15:01 — Status field for the command. Refer to section 3.3.3.2.1. Refer to Figure 97 for values specific to the Connect command. 00 — Reserved

6.5 Property Get Command and Response

The Property Get command is used to specify the property value to return to the host (refer to section 3.1.3). The fields for the Property Get command are defined in Figure 386. If an invalid property or invalid offset is specified, then a status code of Invalid Field in Command shall be returned.

Figure 386: Property Get Command

Bytes	Description
00	**Opcode (OPC):** Set to 7Fh to indicate a Fabrics command.
01	Reserved
03:02	**Command Identifier (CID):** This field specifies a unique identifier for the command. Refer to the definition in Figure 80.
04	**Fabrics Command Type (FCTYPE):** Set to 04h to indicate a Property Get command.
39:05	Reserved
40	**Attributes (ATTRIB):** Specifies attributes for the Property Get command. Bits 7:3 are reserved. Bits 2:0 specifies the size of the property to return. Valid values are shown in the table below. **Value** / **Definition** 000b: 4 bytes 001b: 8 bytes 010b to 111b: Reserved
43:41	Reserved
47:44	**Offset (OFST):** Specifies the offset to the property to get. Refer to section 3.1.3.
63:48	Reserved

The Property Get response is used to return the value of the property requested to the host. The Property Get response is defined in Figure 387.

Figure 387: Property Get Response

Bytes	Description
07:00	**Value (VALUE):** Indicates the value returned for the property if the Property Get command is successful. If the size of the property is four bytes, then the value is specified in bytes 03:00 and bytes 07:04 are reserved.
09:08	**SQ Head Pointer (SQHD):** Indicates the current Submission Queue Head pointer for the associated Submission Queue.
11:10	Reserved
13:12	**Command Identifier (CID):** Indicates the identifier of the command that is being completed.
15:14	**Status (STS):** Indicates status for the command. **Bits** / **Definition** 15:01: Status field for the command. Refer to section 3.3.3.2.1. 00: Reserved

6.6 Property Set Command and Response

The Property Set command is used to set the value of a property (refer to section 3.1.3). The fields for the Property Set command are defined in Figure 388. If an invalid property or invalid offset is specified, then a status code of Invalid Field in Command shall be returned.

Figure 388: Property Set Command

Bytes	Description
00	**Opcode (OPC):** Set to 7Fh to indicate a Fabrics command.
01	Reserved
03:02	**Command Identifier (CID):** This field specifies a unique identifier for the command. Refer to the definition in Figure 80.

Figure 388: Property Set Command

Bytes	Description
04	**Fabrics Command Type (FCTYPE):** Cleared to 00h to indicate a Property Set command.
39:05	Reserved
40	**Attributes (ATTRIB):** Specifies attributes for the Property Set command. Bits 7:3 are reserved. Bits 2:0 specifies the size of the property to update. Valid values are shown in the table below. **Value** / **Definition** 000b: 4 bytes 001b: 8 bytes 010b to 111b: Reserved
43:41	Reserved
47:44	**Offset (OFST):** Specifies the offset to the property to set. Refer to section 3.1.3.
55:48	**Value (VALUE):** Specifes the value used to update the property. If the size of the property is four bytes, then the value is specified in bytes 51:48 and bytes 55:52 are reserved.
63:56	Reserved

The Property Set response provides status for the Property Set command. The Property Set response is defined in Figure 389.

Figure 389: Property Set Response

Bytes	Description
07:00	Reserved
09:08	**SQ Head Pointer (SQHD):** Indicates the current Submission Queue Head pointer for the associated Submission Queue.
11:10	Reserved
13:12	**Command Identifier (CID):** Indicates the identifier of the command that is being completed.
15:14	**Status (STS):** Indicates status for the command. **Bits** / **Definition** 15:01: Status field for the command. Refer to section 3.3.3.2.1. 00: Reserved

7 I/O Commands

An I/O command is a command submitted to an I/O Submission Queue. Figure 390 lists the I/O commands that are defined for use in all I/O Command Sets. The following subsections provide definitions for each command. Refer to section 3.1.2 for mandatory, optional, and prohibited I/O commands for the various controller types. The following subsections describe the definition for each of these commands.

The user data format and any end-to-end protection information is I/O Command Set specific. Refer to each I/O Command Set specification for applicability and additional details, if any. Refer to the referenced I/O Command Set specification for all I/O Command Set specific commands described in Figure 390.

Commands shall only be submitted by the host when the controller is ready as indicated in the Controller Status property (CSTS.RDY) and after appropriate I/O Submission Queue(s) and I/O Completion Queue(s) have been created.

The submission queue entry (SQE) structure and the fields that are common to all I/O commands are defined in section 3.3.3. The completion queue entry (CQE) structure and the fields that are common to all I/O commands are defined in section 3.3.3.2. The command specific fields in the SQE and CQE structures (i.e., SQE Command Dwords 10-15, CQE Dword 0, and CQE Dword 1) for I/O Commands supported across all I/O Command Sets are defined in this section.

Figure 390: Opcodes for I/O Commands

Opcode by Field			Combined Opcode[1]	Command[2]	Reference
(07)	(06:02)	(01:00)			
Standard Command	Function	Data Transfer[3]			
0b	000 00b	00b	00h	Flush[4]	7.1
0b	000 11b	01b	0Dh	Reservation Register	7.3
0b	000 11b	10b	0Eh	Reservation Report	7.5
0b	001 00b	01b	11h	Reservation Acquire	7.2
0b	001 01b	01b	15h	Reservation Release	7.4
Vendor Specific					
1b	n/a	NOTE 3	80h to FFh	Vendor specific	

Notes:
1. Opcodes not listed are I/O Command Set specific or reserved.
2. All I/O commands use the Namespace Identifier (NSID) field. The value FFFFFFFFh is not supported in this field unless footnote 4 in this figure indicates that a specific command does support that value.
3. Indicates the data transfer direction of the command. All options to the command shall transfer data as specified or transfer no data. All commands, including vendor specific commands, shall follow this convention: 00b = no data transfer; 01b = host to controller; 10b = controller to host; 11b = bidirectional.
4. This command may support the use of the Namespace Identifier (NSID) field set to FFFFFFFFh.

7.1 Flush command

The Flush command is used to request that the contents of volatile write cache be made non-volatile.

If a volatile write cache is enabled (refer to section 5.27.1.4), then the Flush command shall commit data and metadata associated with the specified namespace(s) to non-volatile media. The flush applies to all commands for the specified namespace(s) completed by the controller prior to the submission of the Flush command. The controller may also flush additional data and/or metadata from any namespace.

If bits 2:1 are set to 11b in the VWC field (refer to Figure 275) and the specified NSID is FFFFFFFFh, then the Flush command applies to all namespaces attached to the controller processing the Flush command. If bits 2:1 are set to 10b in the VWC field and the specified NSID is FFFFFFFFh, then the controller aborts the command with a status code of Invalid Namespace or Format. If bits 2:1 are cleared to 00b in the VWC field, then the controller behavior if the specified NSID is FFFFFFFFh is not indicated. Controllers compliant

with NVM Express Base Specification revision 1.4 and later shall not set bits 2:1 in the VWC field to the value of 00b.

If a volatile write cache is not present or not enabled, then Flush commands shall have no effect and:

a) shall complete successfully if a sanitize operation is not in progress; and
b) may complete successfully if a sanitize operation is in progress.

All command specific fields are reserved.

7.1.1 Command Completion

Upon completion of the Flush command, the controller posts a completion queue entry to the associated I/O Completion Queue.

7.2 Reservation Acquire command

The Reservation Acquire command is used to acquire a reservation on a namespace, preempt a reservation held on a namespace, and abort a reservation held on a namespace.

The command uses Command Dword 10 and a Reservation Acquire data structure in memory. If the command uses PRPs for the data transfer, then PRP Entry 1 and PRP Entry 2 fields are used. If the command uses SGLs for the data transfer, then the SGL Entry 1 field is used. All other command specific fields are reserved.

Figure 391: Reservation Acquire – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the location of a data buffer where data is transferred from. Refer to Figure 87 for the definition of this field.

Figure 392: Reservation Acquire – Command Dword 10

Bits	Description
31:16	Reserved
15:08	**Reservation Type (RTYPE):** This field specifies the type of reservation to be created. The field is defined in Figure 394.
07:04	Reserved
03	**Ignore Existing Key (IEKEY)**: If this bit is set to a '1', the controller shall return an error of Invalid Field in Command. If this bit is cleared to '0', then the Current Reservation Key is checked.
02:00	**Reservation Acquire Action (RACQA):** This field specifies the action that is performed by the command. RACQA Value / Description / Reference 000b / Acquire / 8.19.5 001b / Preempt / 8.19.7 010b / Preempt and Abort / 8.19.7 011b to 111b / Reserved

Figure 393: Reservation Acquire Data Structure

Bytes	Description
07:00	**Current Reservation Key (CRKEY):** The field specifies the current reservation key associated with the host.
15:08	**Preempt Reservation Key (PRKEY):** If the Reservation Acquire Action is set to 001b (i.e., Preempt) or 010b (i.e., Preempt and Abort), then this field specifies the reservation key to be unregistered from the namespace. For all other Reservation Acquire Action values, this field is reserved.

Figure 394: Reservation Type Encoding

Value	Description
0h	Reserved
1h	Write Exclusive Reservation
2h	Exclusive Access Reservation
3h	Write Exclusive - Registrants Only Reservation
4h	Exclusive Access - Registrants Only Reservation
5h	Write Exclusive - All Registrants Reservation
6h	Exclusive Access - All Registrants Reservation
7h to FFh	Reserved

7.2.1 Command Completion

When the command is completed, the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command.

7.3 Reservation Register command

The Reservation Register command is used to register, unregister, or replace a reservation key.

The command uses Command Dword 10 and a Reservation Register data structure in memory (refer to Figure 397). If the command uses PRPs for the data transfer, then PRP Entry 1 and PRP Entry 2 fields are used. If the command uses SGLs for the data transfer, then the SGL Entry 1 field is used. All other command specific fields are reserved.

Figure 395: Reservation Register – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the location of a data buffer where data is transferred from. Refer to Figure 87 for the definition of this field.

Figure 396: Reservation Register – Command Dword 10

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:30</td><td>Change Persist Through Power Loss State (CPTPL): This field allows the Persist Through Power Loss (PTPL) state associated with the namespace to be modified as a side effect of processing this command. If the Reservation Persistence Feature (refer to section 5.27.1.27) is saveable, then any change to the PTPL state as a result of processing this command shall be applied to both the current value and the saved value of that feature.
<table>
<tr><th>CPTPL Value</th><th>Description</th></tr>
<tr><td>00b</td><td>No change to PTPL state</td></tr>
<tr><td>01b</td><td>Reserved</td></tr>
<tr><td>10b</td><td>Clear PTPL state to ‘0’. Reservations are released and registrants are cleared on a power on.</td></tr>
<tr><td>11b</td><td>Set PTPL state to ‘1’. Reservations and registrants persist across a power loss.</td></tr>
</table></td></tr>
<tr><td>29:04</td><td>Reserved</td></tr>
<tr><td>03</td><td>Ignore Existing Key (IEKEY): If this bit is set to a ‘1’, then Reservation Register Action (RREGA) field values that use the Current Reservation Key (CRKEY) shall succeed regardless of the value of the Current Reservation Key field in the command (i.e., the current reservation key is not checked).</td></tr>
</table>

Figure 396: Reservation Register – Command Dword 10

Bits	Description
02:00	**Reservation Register Action (RREGA):** This field specifies the registration action that is performed by the command. RREGA Value / Description / Reference: 000b – Register Reservation Key – 8.19.3 001b – Unregister Reservation Key – 8.19.4 010b – Replace Reservation Key – 8.19.3 011b to 111b – Reserved

Figure 397: Reservation Register Data Structure

Bytes	Description
07:00	**Current Reservation Key (CRKEY):** If the Reservation Register Action is 001b (i.e., Unregister Reservation Key) or 010b (i.e., Replace Reservation Key), then this field contains the current reservation key associated with the host. For all other Reservation Register Action values, this field is reserved. The controller ignores the value of this field when the Ignore Existing Key (IEKEY) bit is set to '1'.
15:08	**New Reservation Key (NRKEY):** If the Reservation Register Action field is cleared to 000b (i.e., Register Reservation Key) or 010b (i.e., Replace Reservation Key), then this field contains the new reservation key associated with the host. For all other Reservation Register Action values, this field is reserved.

7.3.1 Command Completion

When the command is completed, the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command.

7.4 Reservation Release command

The Reservation Release command is used to release or clear a reservation held on a namespace.

The command uses Command Dword 10 and a Reservation Release data structure in memory. If the command uses PRPs for the data transfer, then PRP Entry 1 and PRP Entry 2 fields are used. If the command uses SGLs for the data transfer, then the SGL Entry 1 field is used. All other command specific fields are reserved.

Figure 398: Reservation Release – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the location of a data buffer where data is transferred from. Refer to Figure 87 for the definition of this field.

Figure 399: Reservation Release – Command Dword 10

Bits	Description
31:16	Reserved
15:08	**Reservation Type (RTYPE):** If the Reservation Release Action field is cleared to 000b (i.e., Release), then this field specifies the type of reservation that is being released. The reservation type in this field shall match the current reservation type. If the reservation type in this field does not match the current reservation type, then the controller should return a status code of Invalid Field in Command. This field is defined in Figure 394.
07:04	Reserved
03	**Ignore Existing Key (IEKEY)**: If this bit is set to a '1', the controller shall return an error of Invalid Field in Command. If this bit is cleared to '0', then the Current Reservation Key is checked.

Figure 399: Reservation Release – Command Dword 10

Bits	Description
02:00	**Reservation Release Action (RRELA):** This field specifies the reservation action that is performed by the command. Value / Description / Reference: 000b – Release – 8.19.6 001b – Clear – 8.19.8 010b to 111b – Reserved

Figure 400: Reservation Release Data Structure

Bytes	O/M	Description
7:0	M	**Current Reservation Key (CRKEY):** The field specifies the current reservation key associated with the host.

7.4.1 Command Completion

When the command is completed, the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command.

7.5 Reservation Report command

The Reservation Report command returns a Reservation Status data structure to memory that describes the registration and reservation status of a namespace.

The size of the Reservation Status data structure is a function of the number of controllers in the NVM subsystem that are associated with hosts that are registrants of the namespace (i.e., there is a Registered Controller data structure and/or Registered Controller extended data structure for each such controller). The controller returns the data structure in Figure 404 if the host has selected a 64-bit Host Identifier and the data structure in Figure 405 if the host has selected a 128-bit Host Identifier (refer to section 5.27.1.25).

If a 64-bit Host Identifier has been specified and the Extended Data Structure bit is set to ‘1’ in Command Dword 11, then the controller shall abort the command with the status code of Host Identifier Inconsistent Format. If a 128-bit Host Identifier has been specified and the Extended Data Structure bit is cleared to ‘0’ in Command Dword 11, then the controller shall abort the command with the status code of Host Identifier Inconsistent Format.

The command uses Command Dword 10 and Command Dword 11. If the command uses PRPs for the data transfer, then PRP Entry 1 and PRP Entry 2 fields are used. If the command uses SGLs for the data transfer, then the SGL Entry 1 field is used. All other command specific fields are reserved.

Figure 401: Reservation Report – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the location of a data buffer where data is transferred to. Refer to Figure 87 for the definition of this field.

Figure 402: Reservation Report – Command Dword 10

Bits	Description
31:00	**Number of Dwords (NUMD):** This field specifies the number of dwords of the Reservation Status data structure to transfer. This is a 0’s based value. If this field corresponds to a length that is less than the size of the Reservation Status data structure, then only that specified portion of the data structure is transferred. If this field corresponds to a length that is greater than the size of the Reservation Status data structure, then the entire contents of the data structure are transferred and no additional data is transferred.

Figure 403: Reservation Report – Command Dword 11

Bits	Description
31:01	Reserved
00	**Extended Data Structure (EDS):** If set to '1', then the controller returns the extended data structure defined in Figure 405. If cleared to '0', then the controller returns the data structure defined in Figure 404.

Figure 404: Reservation Status Data Structure

Bytes	Description
03:00	**Generation (GEN):** This field contains a 32-bit wrapping counter that is incremented any time any one the following occur: • a Reservation Register command completes successfully on any controller associated with the namespace; • a Reservation Release command with Reservation Release Action (RRELA) set to 001b (i.e., Clear) completes successfully on any controller associated with the namespace; and • a Reservation Acquire command with Reservation Acquire Action (RACQA) set to 001b (Preempt) or 010b (Preempt and Abort) completes successfully on any controller associated with the namespace. If the value of this field is FFFFFFFFh, then the field shall be cleared to 0h when incremented (i.e., rolls over to 0h).
04	**Reservation Type (RTYPE):** This field indicates whether a reservation is held on the namespace. A value of 0h indicates that no reservation is held on the namespace. A non-zero value indicates a reservation is held on the namespace and the reservation type is defined in Figure 394.
06:05	**Number of Registered Controllers (REGCTL):** This field indicates the number of controllers that are associated with hosts that are registrants of the namespace. This indicates the number of Registered Controller data structures and/or Registered Controller extended data structures contained in this data structure.
08:07	Reserved
09	**Persist Through Power Loss State (PTPLS):** This field indicates the Persist Through Power Loss State associated with the namespace. PTPLS Value / Description: 0 – Reservations are released and registrants are cleared on a power on. 1 – Reservations and registrants persist across a power loss.
23:10	Reserved
47:24	**Registered Controller Data Structure 0**
	...
24*n+47: 24*(n+1)	**Registered Controller Data Structure n**

Figure 405: Reservation Status Extended Data Structure

Bytes	Description
23:00	Refer to Figure 404 for definition.
63:24	Reserved
127:64	**Registered Controller Extended Data Structure 0**
	...
64*(n+1)+63: 64*(n+1)	**Registered Controller Extended Data Structure n**

Figure 406: Registered Controller Data Structure

Bytes	Description
01:00	**Controller ID (CNTLID):** This field contains the controller ID (i.e., the value of the CNTLID field in the Identify Controller data structure) of the controller whose status is reported in this data structure. If the controller is a dynamic controller (refer to section 3.1.1) that is not associated with a host, then the Controller ID field shall be set to FFFFh.
02	**Reservation Status (RCSTS):** This field indicates the reservation status of the controller described by this data structure. Bits 7:1 are reserved. Bit 0 is set to '1' if the controller is associated with a host that holds a reservation on the namespace.
07:03	Reserved
15:08	**Host Identifier (HOSTID):** This field contains the 64-bit Host Identifier of the controller described by this data structure.
23:16	**Reservation Key (RKEY):** This field contains the reservation key of the host associated with the controller described by this data structure.

Figure 407: Registered Controller Extended Data Structure

Bytes	Description
01:00	**Controller ID (CNTLID):** Refer to Figure 406 for definition.
02	**Reservation Status (RCSTS):** Refer to Figure 406 for definition.
07:03	Reserved
15:08	**Reservation Key (RKEY):** Refer to Figure 406 for definition.
31:16	**Host Identifier (HOSTID):** This field contains the 128-bit Host Identifier of the controller described by this data structure.
63:32	Reserved

7.5.1 Command Completion

When the command is completed, the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command.

8 Extended Capabilities

This section of the document describes extended capabilities that are optional and available for implementation on an NVMe device.

8.1 Asymmetric Namespace Access Reporting

8.1.1 Asymmetric Namespace Access Reporting Overview

Asymmetric Namespace Access (ANA) occurs in environments where namespace access characteristics (e.g., performance or ability to access the media) may vary based on:

- the controller used to access the namespace (e.g., Fabrics); and
- the internal configuration of the NVM subsystem. Asymmetric Namespace Access Reporting is used to indicate to the host information about those access characteristics.

Shared namespaces may be accessed through controllers via multiple PCIe ports or fabric ports (refer to section 2.4.1). The controllers that provide access to a shared namespace may provide identical access characteristics through all controllers (i.e., symmetric access), or may provide different access characteristics through some controllers (i.e., asymmetric access).

Private namespaces are accessed by only one controller at a time. The access characteristics of the namespace through that controller may be impacted as a result of changes to the internal configuration of the NVM subsystem. If the access characteristics of the namespace through that controller are impacted by the internal configuration of the NVM subsystem, then asymmetric access occurs.

Symmetric access to a namespace occurs when:

- the access characteristics using one controller are identical to the access characteristics when using a different controller; and
- changes to the internal configuration of the NVM subsystem do not impact the access characteristics.

Asymmetric access to a namespace occurs when:

- the access characteristics using one controller may differ from the access characteristics when using a different controller; or
- changes to the internal configuration of the NVM subsystem may impact the access characteristics.

While commands may be sent to a shared namespace through any attached controller with asymmetric access, the characteristics (e.g., performance or ability to access the media) may differ based on which controller is used; as a result, the host should consider those characteristics when selecting which controller to use for each command that accesses the namespace. The NVM subsystem may perform autonomous internal reconfiguration that results in a change to the access characteristics.

If an NVM subsystem supports Asymmetric Namespace Access Reporting, then all controllers in that NVM subsystem shall:

- set bit 3 to ‘1’ in the Controller Multi-path I/O and Namespace Sharing Capabilities (CMIC) field in the Identify Controller data structure (refer to Figure 275) to indicate support for Asymmetric Namespace Access Reporting;
- set bit 0 to ‘1’ in the Asymmetric Namespace Access Capabilities (ANACAP) field in the Identify Controller data structure to indicate that the ANA Optimized state is able to be reported;
- set bit 1 to ‘1’ in the ANACAP field in the Identify Controller data structure if ANA Non-Optimized state is able to be reported;
- set bit 2 to ‘1’ in the ANACAP field in the Identify Controller data structure if ANA Inaccessible state is able to be reported;
- set bit 3 to ‘1’ in the ANACAP field in the Identify Controller data structure if ANA Persistent Loss state is able to be reported;

- set bit 4 to ‘1’ in the ANACAP field in the Identify Controller data structure if ANA Change state is able to be reported;
- support Asymmetric Namespace Access Change Notices (refer to section 5.27.1.8); and
- support the Asymmetric Namespace Access log page (refer to section 5.16.1.13).

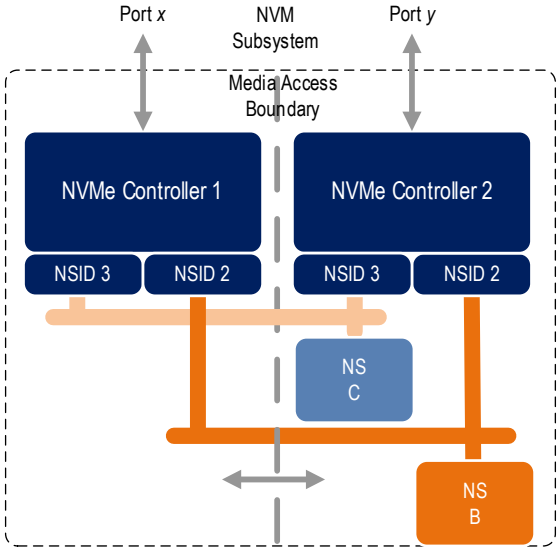
Namespaces attached to a controller that supports Asymmetric Namespace Access Reporting shall:

- be members of an ANA Group; and
- supply a valid ANA Group Identifier in the ANA Group Identifier (ANAGRPID) field in the Identify Namespace data structure (refer to the applicable I/O Command Set specification).

A controller that supports Asymmetric Namespace Access Reporting may also support multiple domains (refer to section 3.2.4).

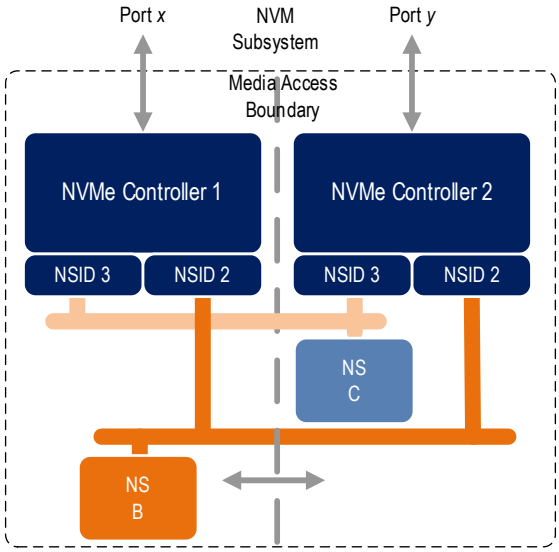
Figure 408 shows an example of an NVM subsystem where access characteristics vary as a result of the presence of two independent domains. In this example, the non-volatile media for namespace B and for namespace C are contained within the same domain that contains controller 2. As a result, controller 2 provides optimized access to namespace B and to namespace C while controller 1 does not provide optimized access to namespace B or to namespace C. In an NVM subsystem that supports multiple domains (refer to section 3.2.4), the Media Access Boundary shown in Figure 408 may be a Communication boundary as shown in Figure 75 and Figure 76.

Figure 408: Namespace B and C optimized through Controller 2



To provide optimized access to namespace B through controller 1, the NVM subsystem may be administratively reconfigured, or may perform autonomous internal reconfiguration actions that change the access characteristics of namespace B when accessed through controller 1 and controller 2 as shown in Figure 409. Controller 2 provides optimized access to namespace C while controller 1 provides optimized access to namespace B. In an NVM subsystem that supports multiple domains (refer to section 3.2.4), the Media Access Boundary shown in Figure 409 may be a Communication boundary as shown in Figure 75 and Figure 76.

Figure 409: Namespace B optimized through Controller 1



8.1.2 ANA Groups

Namespaces that are members of the same ANA Group perform identical asymmetric namespace access state transitions. The ANA Group maintains the same asymmetric namespace access state for all namespaces that are members of that ANA Group (i.e., a change in the asymmetric namespace access state of one namespace only occurs as part of a change in the asymmetric namespace access state of all namespaces that are members of that ANA Group). Namespaces that are members of the same ANA Group shall be members of the same domain (refer to section 2.3.1). The method for assigning namespaces to ANA Groups is outside the scope of this specification.

An ANA Group may contain zero or more namespaces, zero or more NVM Sets, or zero or more Endurance Groups. The mapping of namespaces, NVM Sets, and Endurance Groups to ANA Groups is vendor specific.

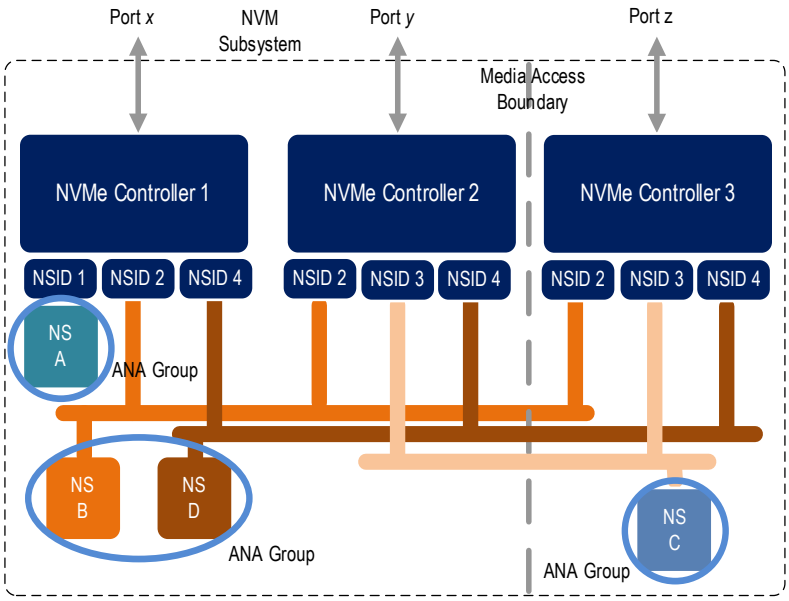
A valid ANA Group Identifier is a non-zero value that is less than or equal to ANAGRPMAX (refer to Figure 275).

The ANA Group Identifier (ANAGRPID) for each ANA Group shall be unique within the NVM subsystem. If bit 6 in the ANACAP field in the Identify Controller data structure is set to ‘1’, then the ANA Group Identifier shall not change while the namespace is attached to any controller in the NVM subsystem. If bit 6 in the ANACAP field is cleared to ‘0’, then the ANA Group Identifier may change while the namespace is attached to any controller in the NVM subsystem. If the ANA Group Identifier changes, the controller shall issue the Asymmetric Namespace Access Change Notice as described in 8.1.3.6.

Figure 410 shows the following four namespaces:

- the private namespace A in a first ANA Group;
- namespace B and namespace D, that are in the same second ANA Group; and
- namespace C that is in a third ANA Group.

Figure 410: Multiple Namespace groups



8.1.3 Asymmetric Namespace Access states

The Asymmetric Namespace Access State indicates information about the characteristics of the relationship between a controller and an ANA Group. The following asymmetric namespace access states are defined:

- ANA Optimized (refer to section 8.1.3.1);
- ANA Non-Optimized (refer to section 8.1.3.2);
- ANA Inaccessible (refer to section 8.1.3.3);
- ANA Persistent Loss (refer to section 8.1.3.4); and
- ANA Change (refer to section 8.1.3.5).

8.1.3.1 ANA Optimized state

While the relationship between the controller and an ANA group is in this state, the characteristic of that relationship to each namespace in that group is optimized. Commands processed by a controller that reports this state for an ANA Group provide optimized access characteristics to any namespace in that ANA Group. A controller that supports ANA Reporting shall support reporting this state.

While in this state, all commands, functions, and operations supported by the namespace shall perform as described in this specification.

8.1.3.2 ANA Non-Optimized state

While the relationship between the controller and an ANA group is in this state, the characteristic of that relationship to each namespace in that group is non-optimized. Commands processed by a controller that reports this state for an ANA Group provide non-optimized access characteristics (e.g., the processing of some commands, especially those involving data transfer, may operate with lower performance or may use NVM subsystem resources less effectively than if a controller is used that reports the optimized state) to any namespace in that ANA Group. Support for reporting this state is optional.

While in this state, all commands, functions, and operations supported by the namespace shall perform as described in this specification.

8.1.3.3 ANA Inaccessible state

While the relationship between the controller and an ANA group is in this state, the characteristic of that relationship to each namespace in that group is inaccessible. Commands processed by a controller that reports this state for an ANA Group are not able to access user data of namespaces in that ANA Group. The namespaces may become accessible through the controller reporting this state at a future time (i.e., a subsequent ANA state transition may occur). Support for reporting this state is optional.

While in this state, accurate namespace related capacity information may not be available. As a result, some namespace capacity information returned in the Identify Namespace data structure (e.g., the NUSE field and the NVMCAP field), are cleared to 0h. For that namespace capacity information, hosts should use the Identify Namespace data structure returned from a controller that reports the relationship between the controller and the namespace to be in the ANA Optimized state or in the ANA Non-Optimized state.

A controller shall abort commands, other than those described in section 8.1.4, with a status code of Asymmetric Access Inaccessible if those commands are submitted while the relationship between the namespace specified by the command and the controller processing the command is in this state.

While ANA Inaccessible state is reported by a controller for the namespace, the host should retry the command on a different controller that is reporting ANA Optimized state or ANA Non-Optimized state. If no controllers are reporting ANA Optimized state or ANA Non-Optimized state, then a transition may be occurring such that a controller reporting the Inaccessible state may become accessible and the host should retry the command on the controller reporting Inaccessible state for at least ANATT seconds (refer to Figure 275). Refer to section 8.10.2.

8.1.3.4 ANA Persistent Loss state

While the relationship between the controller and an ANA group is in this state, the characteristic of that relationship to each namespace in that group is persistently inaccessible. Commands processed by a controller that reports this state for an ANA Group are persistently not able to access user data of namespaces in that ANA Group. The relationship between a controller and an ANA Group in this state shall not transition to any other ANA state. Support for reporting this state is optional.

While in this state, accurate namespace related capacity information may not be available. As a result, some namespace capacity information returned in the Identify Namespace data structure (e.g., the NUSE field and the NVMCAP field), are cleared to 0h. For that namespace capacity information, hosts should use the Identify Namespace data structure returned from a controller that reports the relationship between the controller and the namespace to be in the ANA Optimized state or in the ANA Non-Optimized state.

A controller shall abort commands, other than those described in section 8.1.4, with a status code of Asymmetric Access Persistent Loss if those commands are submitted while the relationship between the namespace specified by the command and the controller processing the command is in this state.

While ANA Persistent Loss state is reported by a controller for the namespace, the host should retry the command on a different controller that is reporting ANA Optimized state or ANA Non-Optimized state. If no controllers are reporting ANA Optimized state or ANA Non-Optimized state, then a transition may be occurring such that a controller reporting the Inaccessible state may become accessible and the host should retry the command on the controller reporting Inaccessible state for at least ANATT seconds (refer to Figure 275).

8.1.3.5 ANA Change state

The change from one asymmetric namespace access state to another asymmetric namespace access state is called a transition. Transitions may occur in such a way that the ANA Change state is not visible to the host (i.e., the ANA Change state may or may not be reported in the Asymmetric Namespace Access State field in the Asymmetric Namespace Access log page (refer to section 5.16.1.13)). Support for reporting this state is optional.

A controller shall abort commands, other than those described in 8.1.4, with a status code of Asymmetric Access Transition if those commands are submitted while the relationship between the namespace specified by the command and the controller processing the command is in this state.

While ANA Change state is reported by a controller for the namespace, the host should:

a) after a short delay, retry the command on the same controller for at least ANATT (refer to Figure 275) seconds (e.g., if ANATT is 30, perform 3 retries at 10 s intervals, or 10 retries at 3 s intervals); or
b) retry the command on a different controller that is reporting ANA Optimized state or ANA Non-Optimized state.

8.1.3.6 Asymmetric Namespace Access Change Notifications

If Asymmetric Namespace Access Change Notices are enabled on a controller, then an Asymmetric Namespace Access Change Notice shall be sent as described in section 5.27.1.8 by the controllers where the change occurred:

a) if an ANA Group Identifier (refer to Figure 275) changes;
b) if an asymmetric namespace access state transition fails (e.g., a transition begins, but does not complete and the controller returns to the state that existed before the transition began); or
c) upon entry to the following ANA states, unless the state entry is a result of a namespace attachment:

- ANA Optimized State;
- ANA Non-Optimized State;
- ANA Inaccessible State; and
- ANA Persistent Loss State.

8.1.4 Asymmetric Namespace Access States Command Processing Effects

Processing of Admin commands that:

- are not NVM Command Set specific commands; and
- do not use the Namespace Identifier (i.e., Figure 139 – “Namespace Identifier Used” column indicates “No”),

are not affected by ANA states, except as specified in Figure 411.

Figure 411 describes Asymmetric Namespace Access effects on command processing.

Figure 411: ANA effects on Command Processing

Command	ANA State	Effects on command processing
Get Features	ANA Inaccessible, ANA Persistent Loss, or ANA Change	The following feature identifiers are not available[1]: a) Reservation Notification Mask (i.e., 82h); b) Reservation Persistence (i.e., 83h); and c) I/O Command Set specific feature identifiers[2].

Figure 411: ANA effects on Command Processing

Command	ANA State	Effects on command processing
Get Log Page	ANA Inaccessible, ANA Persistent Loss, or ANA Change	The following log pages are affected: a) Error Information (i.e., 01h): The log page is not required to contain entries for namespaces whose relationship to the controller processing the command is in the: a. ANA Inaccessible state (refer to section 8.1.3.3); b. the ANA Persistent Loss state (refer to section 8.1.3.4); or c. the ANA Change state (refer to section 8.1.3.5). The following log pages are not available[1]: a) Media Unit Status log page (refer to section 5.16.1.16); and b) Supported Capacity Configuration List log page (refer to section 5.16.1.17).
Identify	ANA Inaccessible or ANA Persistent Loss	Capacity fields in the Identify Namespace data structure (refer to the applicable I/O Command Set specification) information are cleared to 0h.
Set Features	ANA Inaccessible	The saving of features shall not be supported and the following feature identifiers are not available[1]: a) Reservation Notification Mask (i.e., 82h); b) Reservation Persistence (i.e., 83h); and I/O Command Set specific feature identifiers[2]. If the NSID is set to FFFFFFFFh, then the command shall abort[3] with a status code of Asymmetric Access Inaccessible (refer to section 8.1.3.3).

Figure 411: ANA effects on Command Processing

Command	ANA State	Effects on command processing
	ANA Change	The saving of features shall not be supported and the following feature identifiers are not available[1]: a) Reservation Notification Mask (i.e., 82h); b) Reservation Persistence (i.e., 83h); and c) I/O Command Set specific feature identifiers[2]. If the NSID is set to FFFFFFFFh, then the command shall abort[3] with a status code of Asymmetric Access Transition (refer to section 8.1.3.5).
	ANA Persistent Loss	The command shall abort with a status code of Asymmetric Access Persistent Loss (refer to section 8.1.3.4).

Notes:
1. If the ANA state is ANA Inaccessible State, then commands that use feature identifiers or log pages that are not available shall abort with a status code of Asymmetric Access Inaccessible. If the ANA state is ANA Persistent Loss State, then commands that use feature identifiers or log pages that are not available shall abort with a status code of Asymmetric Access Persistent Loss. If the ANA state is ANA Change State, then commands that use feature identifiers or log pages that are not available shall abort with a status code of Asymmetric Access Transition.
2. I/O Command Set specific definition. Refer to each I/O Command Set specification for applicability and additional details, if any.
3. If any namespace that is attached to the controller is in an ANA Group that is in the ANA Inaccessible state, the ANA Persistent Loss state, or the ANA Change state, then the command shall abort with the indicated status. Depending on the ANA state of the ANA Group that contains a namespace (e.g., an ANA state changes during the processing of the command), the specified feature identifier may be altered for some attached namespaces and not altered for other attached namespaces.

8.2 Boot Partitions

Boot Partitions provide an optional area of NVM storage that may be read by the host without the host initializing queues or enabling the controller. The simplified interface to access Boot Partitions may be used for platform initialization code (e.g., a bootloader that is executed from host ROM) to boot to a pre-OS environment (e.g., UEFI) instead of storing the image on another non-volatile storage medium (e.g., SPI flash). Refer to section 8.2.1 for the procedure to read the contents of a Boot Partition.

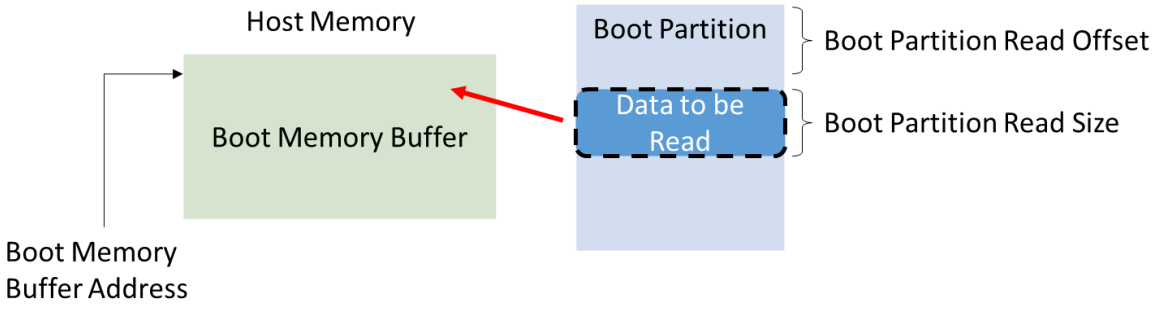
A controller that supports Boot Partitions has two Boot Partitions of equal size using Boot Partition identifiers 0h and 1h. The two Boot Partitions allow the host to update one and verify the contents before marking the Boot Partition active. Controllers in the NVM subsystem may share the same Boot Partitions.

The contents of Boot Partitions are only modified using the Firmware Image Download and Firmware Commit commands (refer to section 8.2.2) and may be secured using Replay Protected Memory Block to prevent unauthorized modifications (refer to section 8.2.3).

8.2.1 Reading from a Boot Partition

A Boot Partition is a continuous block of data as shown in Figure 412, that the host may read via NVMe properties.

Figure 412: Boot Partition Overview



To read the contents of a Boot Partition using NVMe properties, the host allocates a Boot Partition Memory Buffer in host memory for the controller to copy contents from a Boot Partition. The host initializes the Boot Partition Memory Buffer Base Address. The host sets the Boot Partition ID, Boot Partition Read Size, and Boot Partition Read Offset to initiate the Boot Partition read operation. The host may continue reading from the Boot Partition until the entire Boot Partition has been read.

A portion of the Boot Partition may be read by the host any time the NVM subsystem is powered (i.e., whether or not CC.EN is set to '1'). The host shall not modify transport specific properties (described in the applicable NVMe Transport binding specification), reset, or shutdown the controller while a Boot Partition read is in progress.

To read data from a Boot Partition, the host follows these steps:

1. Initialize the transport (e.g., PCIe link), if necessary;
2. Determine if Boot Partitions are supported by the controller (CAP.BPS);
3. Determine which Boot Partition is active (BPINFO.ABPID) and the size of the Boot Partition (BPINFO.BPSZ);
4. Allocate a physically contiguous memory buffer in the host to store the contents of a Boot Partition;
5. Initialize the address (BPMBL.BMBBA) into the memory buffer where the contents should be copied;
6. Initiate the transfer of data from a Boot Partition by writing to the Boot Partition Read Select (BPRSEL) property. This includes setting the Boot Partition identifier (BPRSEL.BPID), size of Boot Partition Read Size (BPRSEL.BPRSZ) and Boot Partition Read Offset (BPRSEL.BPROF). The controller sets the Boot Read Status (BPINFO.BRS) field while transferring the Boot Partition contents to indicate a Boot Partition read operation is in progress; and
7. Wait for the controller to completely transfer the requested portion of the Boot Partition, indicated in the status field (BPINFO.BRS). If BPINFO.BRS is set to 10b, the requested Boot Partition data has been transferred to the Boot Partition Memory Buffer. If BPINFO.BRS is set to 11b, there was an error transferring the requested Boot Partition data and the host may request the Boot Partition data again.

In constrained memory environments, the host may read the contents of a Boot Partition with a small Boot Partition Memory Buffer by reading a small portion of a Boot Partition, moving the data out of the Boot Memory Buffer to another memory location, and then reading another portion of the Boot Partition until the entire Boot Partition has been read.

If the Boot Partition log page is supported (refer to section 5.16.1.1), then the Boot Partition can be accessed through the Boot Partition log page (refer to section 5.16.1.21).

8.2.2 Writing to a Boot Partition

Boot Partition contents may be modified by the host using the Firmware Image Download and Firmware Commit commands while the controller is enabled (CC.EN set to '1').

The process for updating a Boot Partition is:

1. The host issues a Firmware Image Download command to download the contents of the Boot Partition to a controller. There may be multiple portions of the Boot Partition to download, thus the offset for each portion of the Boot Partition being downloaded is specified in the Firmware Image Download command. Host software shall send the Boot Partition image in order starting with the beginning of the Boot Partition;
2. Unlock Boot Partitions for writing (refer to section 8.2.3);
3. The host submits a Firmware Commit command (refer to section 5.12) on that controller with a Commit Action of 110b which specifies that the downloaded image replaces the contents of the Boot Partition specified in the Boot Partition ID field;
4. The controller completes the Firmware Commit command. The following actions are taken in certain error scenarios:
   a. If the firmware activation was not successful because the Boot Partition could not be written, then the controller reports an error of Boot Partition Write Prohibited;
5. (Optional) The host reads the contents of the Boot Partition to verify they are correct (refer to section 8.2.1). Host software updates the active Boot Partition ID by issuing a Firmware Commit command with a Commit Action of 111b; and
6. The host locks Boot Partition access to prevent further modification (refer to section 8.2.3).

If an internal error, reset, or power loss condition occurs while committing the downloaded image to a Boot Partition, the contents of the Boot Partition may contain the old contents, new contents, or a mixture of both. Host software should verify the contents of a Boot Partition before marking that Boot Partition active to ensure the active Boot Partition is stable.

Host software should not read the contents of a Boot Partition while writing to the Boot Partition. The controller may return a combination of new and old data if the host attempts to perform a Boot Partition read operation while overwriting the contents.

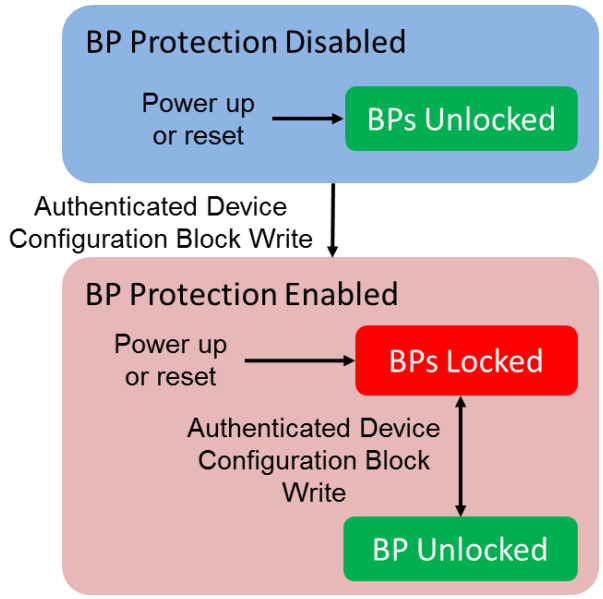
Host software should not overlap firmware/boot partition image update command sequences (refer to section 1.5.23). During a boot partition image update command sequence, if a Firmware Image Download command or a Firmware Commit command is submitted for another firmware/boot partition image update command sequence, the results of both that command and the in-progress firmware image update are undefined.

Host software should use the same controller or Management Endpoint (refer to the NVM Express Management Interface Specification) for all commands that are part of a boot partition image update command sequence. If the commands for a single firmware/boot partition image update command sequence are submitted to more than one controller and/or Management Endpoint, the controller may abort the Firmware Commit command with Invalid Firmware Image status.

8.2.3 Boot Partition Protection

A controller that supports Boot Partitions and RPMB shall support Boot Partition Protection. Boot Partition Protection may be configured using RPMB (refer to section 8.18). Figure 413 shows an overview of Boot Partition Protection.

Figure 413: Boot Partition Protection Overview



The default state for all Boot Partitions is the “Unlocked” state. In this state, host software may read and write a Boot Partition.

All Boot Partitions remain unlocked until Boot Partition Protection is enabled by host software. Host software enables Boot Partition Protection by setting the Boot Partition Protection Enable bit in the RPMB Device Configuration Block data structure (refer to section 8.18). Once Boot Partition Protection is enabled, the controller shall reject Authenticated Device Configuration Block Writes that disable Boot Partition Protection (i.e., enabling Boot Partition Protection is permanent). Once Boot Partition Protection is enabled, Boot Partitions are able to be modified only after unlocking the Boot Partition using RPMB.

After activating Boot Partition Protection, the default state for all Boot Partitions is the “Locked” state. In this state, host software may read a Boot Partition. In this state, the controller rejects attempts to write to a Boot Partition using the Firmware Commit command.

Each Boot Partition may be locked or unlocked independently using the corresponding bit in the Device Configuration Block data structure.

8.3 Capacity Management

8.3.1 Overview

Capacity Management is a capability for organizing physical media into Endurance Groups and NVM Sets. There are two different forms of Capacity Management, Fixed Capacity Management and Variable Capacity Management. A controller that supports Capacity Management shall support at least one form.

The host uses Fixed Capacity Management to create Endurance Groups and NVM Sets by selecting a configuration which explicitly allocates Media Units (refer to section 8.3.2) to Endurance Groups and NVM Sets.

The host uses Variable Capacity Management to:

- create a single Endurance Group by specifying the desired capacity;
- create a single NVM Set by specifying the desired capacity;
- delete a single Endurance Group; and
- delete a single NVM Set.

The Capacity Adjustment Factor is the ratio between the capacity consumed by an Endurance Group from the Unallocated NVM Capacity field in the Identify Controller data structure and the total NVM capacity in that Endurance Group, i.e.:

$$Capacity\ Adjustment\ Factor = INTEGER\left(\frac{Capacity\ Consumed}{Endurance\ Group\ Capacity} * 100\right)$$

(E.g., the value 200 indicates that creation of an Endurance Group with a total NVM capacity of 5 GiB consumes 10 GiB of the Unallocated NVM Capacity indicated by the controller).

If an Endurance Group is created, then the controller performs the following actions as an atomic operation:

a) the value indicated by the Unallocated NVM Capacity (UNVMCAP) field of the Identify Controller data structure is changed based on the requested capacity, the Capacity Adjustment Factor of the created Endurance Group, and the granularity at which the controller allocates NVM capacity; and
b) the Endurance Group Identifier is added to the Endurance Group List.

If an Endurance Group is deleted, then the controller performs the following actions in sequence:

1) the Endurance Group Identifier is removed from the Endurance Group List;
2) if the Media Unit Status log page is supported, then the Endurance Group Identifier field is cleared to 0h in all Media Unit Status Descriptors, if any, that indicate the deleted Endurance Group;
3) every NVM Set in the Endurance Group is deleted; and
4) the value indicated by the Unallocated NVM Capacity (UNVMCAP) field of the Identify Controller data structure is increased by the value that was indicated by the Total Endurance Group Capacity (TEGCAP) field of the Endurance Group Information log page of the deleted Endurance Group.

If any of the entities modified by the above sequence are accessed after the sequence begins and before it completes, then the results are indeterminate.

If an NVM Set is created, then the controller performs the following actions as an atomic operation:

a) the NVM Set Identifier is added to the NVM Set List; and
b) the Unallocated Endurance Group Capacity indicated by the Endurance Group Information log page (refer to Figure 217) is decreased by the amount of capacity allocated to the NVM Set; the controller may allocate NVM capacity in units such that the requested size for an NVM Set may be rounded up to the next unit boundary.

If an NVM Set is deleted, then the controller performs the following actions in sequence:

1) the NVM Set Identifier is removed from the NVM Set List;
2) if the Media Unit Status log page is supported, then the NVM Set Identifier field is cleared to 0h in all Media Unit Status Descriptors, if any, that indicated the deleted NVM Set;
3) for each namespace in the deleted NVM Set:
    a. all commands targeting the namespace are handled as described for namespace deletion in section 8.11;
    b. the namespace identifier is removed from the Allocated Namespace ID list; the namespace is deleted;
    c. the namespace identifier is added to the Changed Namespace List; and
    d. a Namespace Attribute Changed event is generated for hosts other than the host which issued the Capacity Management command;

    and
4) the Unallocated Endurance Group Capacity indicated by the Endurance Group Information log page is increased by the amount of capacity formerly allocated to the NVM Set.

If any of the entities modified by the above sequence are accessed after the sequence begins and before it completes, then the results are indeterminate.

If an NVM Set is created or deleted, the value indicated by the Unallocated NVM Capacity (UNVMCAP) field of the Identify Controller data structure is not changed.

8.3.2 Fixed Capacity Management

A Media Unit represents a component of the underlying media in a domain. An implementation may choose to represent each die as a separate Media Unit; however, this is not required. A Media Unit is the smallest media component for which the controller reports wear information (refer to the Available Spare field and the Percentage Used field in the Media Unit Status Descriptor, Figure 249).

Two or more I/O operations to a Media Unit at the same time may interfere with each other as they contend for resources internal to or shared by that Media Unit.

A controller supporting Fixed Capacity Management:

a) shall support the Media Unit Status log page (refer to section 5.16.1.16);
b) shall support Endurance Groups (refer to section 3.2.3);
c) may support NVM Sets (refer to section 3.2.2);
d) shall set the Fixed Capacity Management bit to ‘1’ in the CTRATT field of the Identify Controller data structure (refer to Figure 275);
e) shall support the Supported Capacity Configuration List log page (refer to section 5.16.1.17); and
f) shall support the Select Capacity Configuration operation of the Capacity Management command (refer to section 5.3).

Media Units are accessed by way of Channels that represent communication pathways that may be a source of contention. This information is reported to facilitate the composition of NVM Sets that minimize interference between independent writers competing for this type of resource.

The host allocates the Media Units in a domain to Endurance Groups and NVM Sets by:

1) reading the Supported Capacity Configuration List log page (refer to Figure 250) and selecting the desired configuration; and
2) issuing a Capacity Management command specifying the Select Capacity Configuration operation and the Capacity Configuration Identifier of the desired configuration.

Following successful completion of the command, each Media Unit is allocated to one Endurance Group and to one NVM Set. The resulting configuration of Media Units is reported by the Media Unit Status log page (refer to section 5.16.1.16).

8.3.3 Variable Capacity Management

Variable Capacity Management allows the dynamic creation and deletion of Endurance Groups and NVM Sets.

A controller supporting Variable Capacity Management:

a) may support the Media Unit Status log page;
b) shall support Endurance Groups;
c) may support NVM Sets;
d) shall set the Variable Capacity Management bit to ‘1’ in the CTRATT field of the Identify Controller data structure (refer to Figure 275);
e) shall support the Create Endurance Group operation of the Capacity Management command;
f) may support the Delete Endurance Group operation of the Capacity Management command; and
g) if NVM Sets are supported:
    a. shall support the Create NVM Set operation of the Capacity Management command;
    b. shall report non-zero values in the Total Endurance Group Capacity field and the Unallocated Endurance Group Capacity field in the Endurance Group Information log page (refer to Figure 217); and
    c. may support the Delete NVM Set operation of the Capacity Management command.

If a controller supports the Delete Endurance Group operation of the Capacity Management command, then it shall set the Delete Endurance Group bit to ‘1’ in the CTRATT field of the Identify Controller data structure.

If a controller supports the Delete NVM Set operation of the Capacity Management command, then it shall set the Delete NVM Set bit to ‘1’ in the CTRATT field of the Identify Controller data structure.

A typical sequence of operations for allocating capacity is:

1) determine the available capacities in each domain (refer to section 3.2.4);
2) create Endurance Group with desired capacity (refer to section 5.3);
3) create NVM Set with desired capacity in the Endurance Group (refer to section 5.3); and
4) create namespace with desired capacity in the NVM Set (refer to section 5.23).

A typical sequence of operations for deallocating capacity is:

1) delete namespace, if any, (refer to section 5.23);
2) delete NVM Set, if any, (refer to section 5.3); and
3) delete Endurance Group (refer to section 5.3).

If there is insufficient unallocated capacity in an Endurance Group for the controller to create an NVM Set, then the host can delete one or more NVM Sets in that Endurance Group and create the new NVM Set using some or all of the available capacity.

If there is insufficient unallocated capacity in a domain for the controller to create an Endurance Group, then the host can delete one or more Endurance Groups in that domain and create a new Endurance Group using some or all of the available capacity.

8.4 Command and Feature Lockdown

The Command and Feature Lockdown capability is used to prohibit the execution of commands submitted to NVM Express controllers and/or Management Endpoints in an NVM subsystem. Within this feature, commands and Feature Identifiers are defined with the following scopes:

- Admin Command Set commands defined by the Opcode field;
- Set Features command features defined by the Feature Identifier field;
- Management Interface Command Set commands defined by the Opcode field (refer to the NVM Express Management Interface Specification); and
- PCIe Command Set commands defined by the Opcode field (refer to the NVM Express Management Interface Specification).

Admin Command Set commands and Feature Identifiers are defined to be prohibitable by this feature, however it is vendor specific which of the Command Set commands and Feature Identifiers are prohibitable from execution, including the Lockdown command itself.

The prohibition of commands or Feature Identifiers on an interface is specified in the Interface field of the Lockdown command (refer to section 5.19). The prohibition applies to all applicable:

- NVM Express controllers; and
- Management Endpoints,

in the NVM subsystem, as specified in the Interface field.

The Lockdown command is used to specify commands that are prohibited from execution (i.e., locked down) and may be used further to then again allow that command to be executed.

The prohibiting of execution of a command as part of this feature shall persist until:

a) power cycle of the NVM subsystem; or
b) further being allowed by a follow-on Lockdown command.

If a prohibited Admin Command Set command or Feature Identifier is processed by a controller in the NVM subsystem, then that command shall be aborted with a status code of Command Prohibited by Command and Feature Lockdown.

If a prohibited Management Interface Command Set command or PCIe Command Set command is processed by a management endpoint in the NVM subsystem, then that command shall be aborted and send a Response Message with an Access Denied Error Response (refer to the NVM Express Management Interface Specification).

The prohibition or allowance of the execution of commands using this feature treats from where the command was received, being an NVM Express controller Admin Submission Queue or an out-of-band Management Endpoint independently. This means that a command may be prohibited if received on an NVM Express controller Admin Submission Queue but allowed if received on an out-of-band Management Endpoint, if supported. The Interface field in the Lockdown command is used to specify this behavior.

A host may use the Command and Feature Lockdown log page (refer to section 5.16.1.20) to determine the commands and Feature Identifiers that are allowed to be prohibited from execution. A Get Log Page command specifying the Command and Feature Lockdown log page returns a list of command opcodes or Feature Identifiers depending on the scope specified in the Get Log Page command. The returned list of opcodes or Feature Identifiers are the opcodes or Feature Identifiers that are:

a) supported as being prohibitable from execution using the Lockdown command;
b) currently prohibited from execution if received on an NVM Express controller submission queue; or
c) currently prohibited from execution if received out-of-band on a Management Endpoint.

If the Command and Feature Lockdown capability is supported (i.e., bit 10 in the OACS field in Figure 275 is set to ‘1’), then the controller shall support the Lockdown command and the Command and Feature Lockdown log.

8.5 Controller Memory Buffer

The Controller Memory Buffer (CMB) is a region of general purpose read/write memory on the controller. The controller indicates support for the CMB by setting CAP.CMBS to ‘1’. The host indicates intent to use the CMB by setting CMBMSC.CRE to ‘1’. Once this bit is set to ‘1’, the controller indicates the properties of the CMB via the CMBLOC and CMBSZ properties (refer to section 3.1.3).

The CMB may be used for a variety of purposes. The controller indicates which purposes the memory may be used for by setting support flags in the CMBSZ property.

The CMB’s PCI Express address range is used for external memory read and write requests to the CMB. The PCI Express base address of the CMB is defined by the PCI Base Address Register (BAR) indicated by CMBLOC.BIR, and the offset indicated by CMBLOC.OFST. The size of the CMB is indicated by CMBSZ.SZ.

The controller uses the CMB’s controller address range to reference CMB with addresses supplied by the host. The PCI Express address range and the controller address range of the CMB may differ, but both ranges have the same size, and equivalent offsets within each range have a one-to-one correspondence. The host configures the controller address range via the CMBMSC property.

The host enables the CMB’s controller memory space via the CMBMSC.CMSE bit. When controller memory space is enabled, if host supplies an address referencing the CMB’s controller address range, then the controller directs memory read or write requests for this address to the CMB.

When the CMB’s controller memory space is disabled, the controller does not consider any host-supplied address to reference the CMB’s controller address range, and memory read and write requests are directed elsewhere (e.g., to memory other than the CMB).

To prevent possible misdirection of the controller’s memory requests, before host software enables the CMB’s controller memory space, the host should configure the CMB’s controller address range to so that the addresses do not overlap any address that host software intends to use for DMA.

In versions prior to NVM Express Base Specification revision 1.4, for a controller that supports the CMB, the CMB’s controller address range is fixed to be equal to its PCI Express address range, and the CMB’s controller memory space is always enabled whenever the controller is enabled. To prevent misdirection of controller memory requests when such a controller is assigned to a virtual machine, host software (on the

hypervisor or host OS) should not enable translation of the CMB's PCI Express address range and should ensure that this address range does not overlap any range of pre-translated addresses that the virtual machine may use for DMA.

Host software may configure the CMBMSC property so that CMB operates when the controller is assigned to a virtual machine that only supports NVM Express Base Specification revision 1.3 and earlier. To prevent that virtual machine from unintentionally clearing the CMBMSC property to 0h, the contents of the CMBSMC property are preserved across Controller Reset and Function Level Reset.

Submission Queues in host memory require the controller to perform a PCI Express read from host memory in order to fetch the submission queue entries. Submission Queues in controller memory enable host software to directly write the entire submission queue entry to the controller's internal memory space, avoiding one read from the controller to the host. This approach reduces latency in command execution and improves efficiency in a PCI Express fabric topology that may include multiple switches. Similarly, PRP Lists or SGLs require separate fetches across the PCI Express fabric, which may be avoided by writing the PRP or SGL to the Controller Memory Buffer. Completion Queues in the Controller Memory Buffer may be used for peer to peer or other applications. For writes of small amounts of data, it may be advantageous to have the host write the data and/or metadata to the Controller Memory Buffer rather than have the controller fetch it from host memory.

The contents of the Controller Memory Buffer are undefined as the result of:

- the CMBMSC.CMSE bit transitioning from '0' to '1';
- a Controller Reset; or
- a Function Level Reset.

Host software should initialize any memory in the Controller Memory Buffer before being referenced (e.g., a Completion Queue shall be initialized by host software in order for the Phase Tag to be used correctly).

A CMB implementation has a maximum sustained write throughput. The CMB implementation may also have an optional write elasticity buffer used to buffer writes from CMB PCIe write requests. When the CMB sustained write throughput is less than the PCI Express link throughput, then such a write elasticity buffer allows PCIe write request burst throughput to exceed the CMB sustained write throughput without back pressuring into the PCI Express fabric.

The time required to transfer data from the write elasticity buffer to the CMB is the amount of data written to the elasticity buffer divided by the Controller Memory Buffer Sustained Write Throughput (refer to section 3.1.3.18). The time to transfer the entire contents of the write elasticity buffer is the Controller Memory Buffer Elasticity Buffer Size (refer to section 3.1.3.17) divided by the Controller Memory Buffer Sustained Write Throughput.

A controller memory-based queue is used in the same manner as a host memory-based queue – the difference is the memory address used is located within the controller's own memory rather than in the host memory. The Admin or I/O Queues may be placed in the Controller Memory Buffer. If the CMBLOC.CQMMS bit (refer to Figure 52) is cleared to '0', then for a particular queue, all memory associated with it shall reside in either the Controller Memory Buffer or outside the Controller Memory Buffer.

If the CMBLOC.CQPDS bit (refer to Figure 52) is cleared to '0', then for all queues in the Controller Memory Buffer, the queue shall be physically contiguous.

The controller may support PRP Lists and SGLs in the Controller Memory Buffer. If the CMBLOC.CDPMLS bit (refer to Figure 52) is cleared to '0', then for a particular PRP List or SGL associated with a single command, all memory associated with the PRP List or SGL shall be either entirely located in the Controller Memory Buffer or entirely located outside the Controller Memory Buffer.

PRP Lists and SGLs associated with a command may be placed in the Controller Memory Buffer if that command is present in a Submission Queue in the Controller Memory Buffer. If:

a) CMBLOC.CDPCILS bit (refer to Figure 52) is cleared to '0'; and
b) a command is not present in a Submission Queue in the Controller Memory Buffer,

then the PRP Lists and SGLs associated with that command shall not be placed in the Controller Memory Buffer.

The controller may support data and metadata in the Controller Memory Buffer. If the CMBLOC.CDMMMS bit (refer to Figure 52) is cleared to '0', then all data and metadata, if any, associated with a particular command shall be either entirely located in the Controller Memory Buffer or entirely located outside the Controller Memory Buffer.

If the requirements for the Controller Memory Buffer use are violated by the host, the controller shall abort the associated command with a status code of Invalid Use of Controller Memory Buffer.

The address region allocated for the CMB shall be 4 KiB aligned. It is recommended that a controller allocate the CMB on an 8 KiB boundary. The controller shall support burst transactions up to the maximum payload size, support byte enables, and arbitrary byte alignment. The host shall ensure that all writes to the CMB that are needed for a command have been sent before updating the SQ Tail doorbell property. The Memory Write Request to the SQ Tail doorbell property shall not have the Relaxed Ordering bit set to '1', to ensure that prior writes to the CMB have completed.

8.6 Device Self-test Operations

A device self-test operation is a diagnostic testing sequence that tests the integrity and functionality of the controller and may include testing of the media associated with namespaces. The operation is broken down into a series of segments, where each segment is a set of vendor specific tests. The segment number in the Self-test Result Data Structure (refer to section 5.16.1.7) is used for reporting purposes to indicate where a test failed, if any. The test performed in each segment may be the same for the short device self-test operation and the extended device self-test operation.

A device self-test operation is performed in the background allowing concurrent processing of some commands and requiring suspension of the device self-test operation to process other commands. Which commands may be processed concurrently versus require suspension of the device self-test operation is vendor specific.

If the controller receives any command that requires suspension of the device self-test operation to process and complete, then the controller shall:

1) suspend the device self-test operation;
2) process and complete that command; and
3) resume the device self-test operation.

During a device self-test operation, the performance of the NVM subsystem may be degraded (e.g., controllers not performing the device self-test operation may also experience degraded performance).

The following device self-test operations are defined:

a) short device self-test operation (refer to section 8.6.1); and
b) extended device self-test operation (refer to section 8.6.2).

Figure 414 is an informative example of a device self-test operation with the associated segments and tests performed in each segment.

Figure 414: Example Device Self-test Operation (Informative)

Segment	Test Performed	Failure Criteria
1 – RAM Check	Write a test pattern to RAM, followed by a read and compare of the original data.	Any uncorrectable error or data miscompare
2 – SMART Check	Check SMART or health status for Critical Warning bits set to '1' in SMART / Health Information Log.	Any Critical Warning bit set to '1' fails this segment
3 – Volatile memory backup	Validate volatile memory backup solution health (e.g., measure backup power source charge and/or discharge time).	Significant degradation in backup capability

Figure 414: Example Device Self-test Operation (Informative)

Segment		Test Performed	Failure Criteria
4 – Metadata validation		Confirm/validate all copies of metadata.	Metadata is corrupt and is not recoverable
5 – NVM integrity		Write/read/compare to reserved areas of each NVM. Ensure also that every read/write channel of the controller is exercised.	Data miscompare
Extended only	6 – Data Integrity	Perform background housekeeping tasks, prioritizing actions that enhance the integrity of stored data. Exit this segment in time to complete the remaining segments and meet the timing requirements for extended device self-test operation indicated in the Identify Controller data structure.	Metadata is corrupt and is not recoverable
7 – Media Check		Perform random reads from every available good physical block. Exit this segment in time to complete the remaining segments. The time to complete is dependent on the type of device self-test operation.	Inability to access a physical block
8 – Drive Life		End-of-life condition: Assess the drive's suitability for continuing write operations.	The Percentage Used is set to 255 in the SMART / Health Information Log or an analysis of internal key operating parameters indicates that data is at risk if writing continues
9 – SMART Check		Same as 2 – SMART Check	

8.6.1 Short Device Self-Test Operation

A short device self-test operation should complete in two minutes or less. The percentage complete of the short device self-test operation is indicated in the Current Percentage Complete field in the Device Self-test Log (refer to section 5.16.1.7).

A short device self-test operation:

a) shall be aborted by any Controller Level Reset that affects the controller on which the device self-test is being performed;
b) shall be aborted by a Format NVM command as described in Figure 415;
c) shall be aborted when a sanitize operation is started (refer to section 5.24);
d) shall be aborted if a Device Self-test command with the Self-Test Code field set to Fh is processed; and
e) may be aborted if the specified namespace is removed from the namespace inventory.

Figure 415: Format NVM command Aborting a Device Self-Test Operation

FNA bit [1]	NSID in Format NVM command	NSID in Device Self-test command	Abort Device Self-Test operation?
0	Any allocated NSID value (refer to section 3.2.1.3)	Any active NSID value (refer to section 3.2.1.2)	Yes, if the NSID values are the same
0	FFFFFFFFh	Any active NSID value (refer to section 3.2.1.2)	Yes
0	Any allocated NSID value (refer to section 3.2.1.3)	FFFFFFFFh	Optional
0	FFFFFFFFh	FFFFFFFFh	Yes
1	Ignored	Ignored	Yes
Key: Optional = The device self-test operation is not required to be aborted but may be aborted.			

Figure 415: Format NVM command Aborting a Device Self-Test Operation

FNA bit [1]	NSID in Format NVM command	NSID in Device Self-test command	Abort Device Self-Test operation?

Notes:
1. For a Format NVM command with Secure Erase, this column refers to Bit 1 in the FNA field in the Identify Controller data structure (refer to Figure 275) and bit 0 in the FNA field is ignored. For a Format NVM command without Secure Erase, this column refers to bit 0 in the FNA field, and bit 1 in the FNA field is ignored.

8.6.2 Extended Device Self-Test Operation

An extended device self-test operation should complete in the time indicated in the Extended Device Self-test Time field in the Identify Controller data structure or less. The percentage complete of the extended device self-test operation is indicated in the Current Percentage Complete field in the Device Self-test Log (refer to section 5.16.1.7).

An extended device self-test operation shall persist across any Controller Level Reset and shall resume after completion of the reset or any restoration of power, if any. The segment where the extended device self-test operation resumes is vendor specific, but implementations should only have to perform tests again within the last segment that was being tested prior to the reset.

An extended device self-test operation:

a) shall be aborted by a Format NVM command as described in Figure 415;
b) shall be aborted when a sanitize operation is started (refer to section 5.24);
c) shall be aborted if a Device Self-test command with the Self-Test Code field set to Fh is processed; and
d) may be aborted if the specified namespace is removed from the namespace inventory.

8.7 Directives

Directives is a mechanism to enable host and NVM subsystem or controller information exchange. The Directive Receive command (refer to section 5.10) is used to transfer data related to a specific Directive Type from the controller to the host. The Directive Send command (refer to section 5.11) is used to transfer data related to a specific Directive Type from the host to the controller. Other commands may include a Directive Specific value specific for a given Directive Type (e.g., the Write command in the NVM Command Set).

Support for Directives is optional and is indicated in the Optional Admin Command Support (OACS) field in the Identify Controller data structure (refer to Figure 275).

If a controller supports Directives, then the controller shall:

- Indicate support for Directives in the Optional Admin Command Support (OACS) field in the Identify Controller data structure;
- Support the Directive Receive command;
- Support the Directive Send command; and
- Support the Identify Directive (i.e., Type 00h).

The Directive Types that may be supported by a controller (refer to Figure 416) are the Identify Directive (refer to section 8.7.2), and the Streams Directive (refer to section 8.7.3). The Directive Specific field and Directive Operation field are dependent on the Directive Type specified in the command (e.g., Directive Send, Directive Receive, or I/O command).

Figure 416: Directive Types

Directive	Directive Type Value	Definition	I/O Command Directive
Identify	00h	Section 8.7.2	No
Streams	01h	Section 8.7.3	Yes

If a Directive is not supported or is supported and disabled, then all Directive Send commands and Directive Receive commands with that Directive Type shall be aborted with a status code of Invalid Field in Command.

Support for a specific directive type is indicated using the Return Parameters operation of the Identify Directive. A specific directive may be enabled or disabled using the Enable operation of the Identify Directive. Before using a specific directive, the host should determine if that directive is supported and should enable that directive using the Identify Directive.

8.7.1 Directive Use in I/O Commands

I/O Command Directives are the subset of Directive Types that may be used as part of I/O commands. For example, a Write command in the NVM Command Set may specify a Directive Type and an associated Directive Specific value. I/O Command Directives shall have a Directive Type value that is less than or equal to 0Fh due to the size of the Directive Type field in I/O commands. When a Directive Type is specified in an I/O command, the most significant four bits are assumed to be 0h. A Directive Type of 00h in an I/O command specifies that the I/O command is not using Directives.

In an I/O command, if the Directive Type (DTYPE) field is set to an I/O Command Directive, then the Directive Specific (DSPEC) field includes additional information for the associated I/O command (refer to Figure 417).

Figure 417: Directive Specific Field Interpretation

Directive Type Value	Directive Specific Field Definition
00h (Directives not in use)	Field not used.
01h (Streams)	Specifies the identifier of the stream associated with the data.
02h to 0Fh	Reserved

In an I/O command:

- if no I/O Command Directive is enabled or the DTYPE field is cleared to 00h, then the DTYPE field and the DSPEC field are ignored; and
- if one or more I/O Command Directives is enabled and the DTYPE field is set to a value that is not supported or not enabled, then the controller shall abort the command with a status code of Invalid Field in Command**.**

For the Streams Directive (i.e., DTYPE field set to 01h), if the DSPEC field is cleared to 0h in an I/O command that supports the Streams Directive, then that I/O command shall be processed normally (i.e., as if DTYPE field is cleared to 00h).

8.7.2 Identify (Directive Type 00h)

The Identify Directive is used to determine the Directive Types that the controller supports and to enable use of the supported Directives. If Directives are supported, then this Directive Type shall be supported.

The Directive operations that shall be supported for the Identify Directive are listed in Figure 418.

Figure 418: Identify Directive – Directive Operations

Directive Command	Directive Operation Name	Directive Operation Value	Definition
Directive Receive	Return Parameters	01h	Section 8.7.2.1.1
	Reserved	All other values	
Directive Send	Enable Directive	01h	Section 8.7.2.2.1
	Reserved	All other values	

8.7.2.1 Directive Receive

This section defines operations used with the Directive Receive command for the Identify Directive.

8.7.2.1.1 Return Parameters (Directive Operation 01h)

This operation returns a data structure that contains a bit vector specifying the Directive Types supported by the controller and a bit vector specifying the Directive Types enabled for the namespace. The data structure returned is defined in Figure 419. If an NSID value of FFFFFFFFh is specified, then the controller shall abort the command with a status code of Invalid Field in Command. The DSPEC field in command Dword 11 is not used for this operation.

Figure 419: Identify Directive – Return Parameters Data Structure

Bytes	Bits	Description
		Directives Supported
31:00	255:02	Reserved
	01	**Streams Directive:** This bit is set to '1' if the Streams Directive is supported. This bit is cleared to '0' if the Streams Directive is not supported.
	00	**Identify Directive:** This bit shall be set to '1' to indicate that the Identify Directive is supported.
		Directives Enabled
63:32	255:02	Reserved
	01	**Streams Directive:** This bit is set to '1' if the Streams Directive is enabled. This bit is cleared to '0' if the Streams Directive is not enabled.
	00	**Identify Directive:** This bit shall be set to '1' to indicate that the Identify Directive is enabled.
4095:64	n/a	Reserved

8.7.2.2 Directive Send

This section defines operations used with the Directive Send command for the Identify Directive.

8.7.2.2.1 Enable Directive (Directive Operation 01h)

The Enable Directive operation is used to enable a specific Directive for use within a namespace by all controllers that are associated with the same Host Identifier. The DSPEC field in command Dword 11 is not used for this operation. The Identify Directive is always enabled. The enable state of each Directive on each shared namespace attached to enabled controllers associated with the same non-zero Host Identifier is the same. If an NSID value of FFFFFFFFh is specified, then the Enable Directive operation applies to the NVM subsystem (i.e., all namespaces and all controllers associated with the NVM subsystem). On an NVM Subsystem Reset, all Directives other than the Identify Directive are disabled in the Domains impacted by that reset (refer to section 3.7.1).

On a Controller Level Reset:

- all Directives other than the Identify Directive are disabled for that controller; and
- if there is an enabled controller associated with the Host Identifier for the controller that was reset, then for namespaces attached to enabled controllers associated with that Host Identifier, Directives are not disabled.

If a host sets the Host Identifier of a controller to the same non-zero Host Identifier as one or more other controllers in the NVM subsystem, then setting that Host Identifier shall result in each shared namespace attached to that controller having the same enable state for each Directive as the enable state for each Directive for that namespace attached to other controllers associated with that Host Identifier.

If a host enables a controller that has the same non-zero Host Identifier as one or more other controllers in the NVM subsystem, then enabling that controller shall result in each shared namespace attached to that controller having the same enable state for each Directive as the enable state for each Directive for that namespace attached to other controllers associated with that Host Identifier.

For all controllers in an NVM subsystem that have the same non-zero Host Identifier, if a host changes the enable state of any Directive for a shared namespace attached to a controller by a means other than a Controller Level Reset, then that change shall be made to the enable state of that Directive for that namespace attached to any other controller associated with that Host Identifier.

Figure 420: Enable Directive – Command Dword 12

Bits	Description
31:16	Reserved
15:08	**Directive Type (DTYPE):** This field specifies the Directive Type to enable or disable. If this field specifies the Identify Directive (i.e., 00h), then a status code of Invalid Field in Command shall be returned.
07:01	Reserved
00	**Enable Directive (ENDIR):** If set to '1' and the Directive Type is supported, then the Directive is enabled. If cleared to '0', then the Directive is disabled. If this bit is set to '1' for a Directive that is not supported, then a status code of Invalid Field in Command shall be returned.

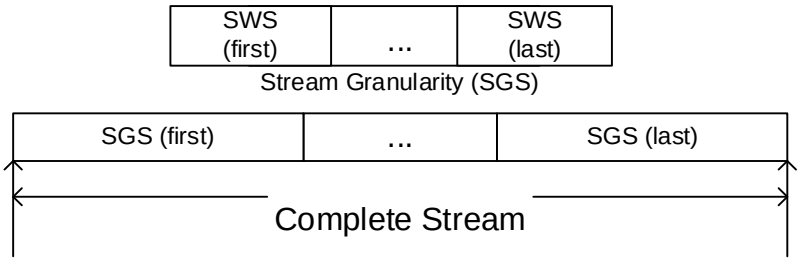
8.7.3 Streams (Directive Type 01h, Optional)

The Streams Directive enables the host to indicate (i.e., by using the stream identifier) to the controller that the specified user data in a User Data Out command (e.g., logical blocks in a write command) are part of one group of associated data. This information may be used by the controller to store related data in associated locations or for other performance enhancements.

The controller provides information in response to the Return Parameters operation about the configuration of the controller that indicates Stream Write Size, Stream Granularity Size, and stream resources at the NVM subsystem and namespace levels.

Data that is aligned to and in multiples of the Stream Write Size (SWS) provides optimal performance of the write commands to the controller. The SWS unit of granularity is defined independently for each I/O Command Set. The Stream Granularity Size indicates the size of the media that is prepared as a unit for future allocation for write commands and is a multiple of the Stream Write Size. The controller may allocate and group together a stream in Stream Granularity Size (SGS) units. Refer to Figure 421.

Figure 421: Directive Streams – Stream Alignment and Granularity



One example of this is if the host issues an NVM Command Set Dataset Management command (refer to the Dataset Management command section of the NVM Command Set Specification) to deallocate logical blocks that are associated with a stream, that host should specify a starting LBA and length that is aligned to and in multiples of the Stream Granularity Size. This provides optimal performance and endurance of the media.

Stream resources are the resources in the NVM subsystem that are necessary to track operations associated with a specified stream identifier. There are a maximum number of stream resources that are available in an NVM subsystem as indicated by the Max Stream Limit (MSL) field in the Return Parameters data structure (refer to Figure 425).

Available NVM subsystem stream resources are stream resources that are not allocated for exclusive use in any namespace. Available NVM subsystem stream resources are reported in the NVM Subsystem Streams Available (NSSA) field (refer to Figure 425) and may be used by any host in any namespace that:

- has the Streams Directive enabled;
- has not been allocated exclusive stream resources by that host if bit 0 of the NSSC field is cleared to '0'; and

- has not been allocated exclusive stream resources by any host if bit 0 of the NSSC field is set to '1'.

Each time stream resources are allocated for exclusive use in a specified namespace, the available NVM subsystem stream resources reported in the NSSA field are reduced.

For a given namespace:

a) a host allocates stream resources to that namespace for the exclusive use of that host(s) by issuing the Allocate Resources operation;
b) other hosts may concurrently allocate stream resources to that namespace for their exclusive use; and
c) hosts which have not allocated stream resources to that namespace may use available NVM subsystem stream resources for access to that namespace.

The Directive operations that shall be supported if the Streams Directive is supported are listed in Figure 422. The Directive Specific field in a command is referred to as the Stream Identifier when the Directive Type field is set to the Streams Directive.

Figure 422: Streams – Directive Operations

Directive Command	Directive Operation Name	Directive Operation Value	Definition
Directive Receive	Return Parameters	01h	Section 8.7.3.1.1
	Get Status	02h	Section 8.7.3.1.2
	Allocate Resources	03h	Section 8.7.3.1.3
	Reserved	All other values	
Directive Send	Release Identifier	01h	Section 8.7.3.2.1
	Release Resources	02h	Section 8.7.3.2.2
	Reserved	All other values	

Stream identifiers are assigned by the host and may be in the range 0001h to FFFFh. The host may specify a sparse set of stream identifiers (i.e., there is no requirement for the host to use Stream Identifiers in any particular order).

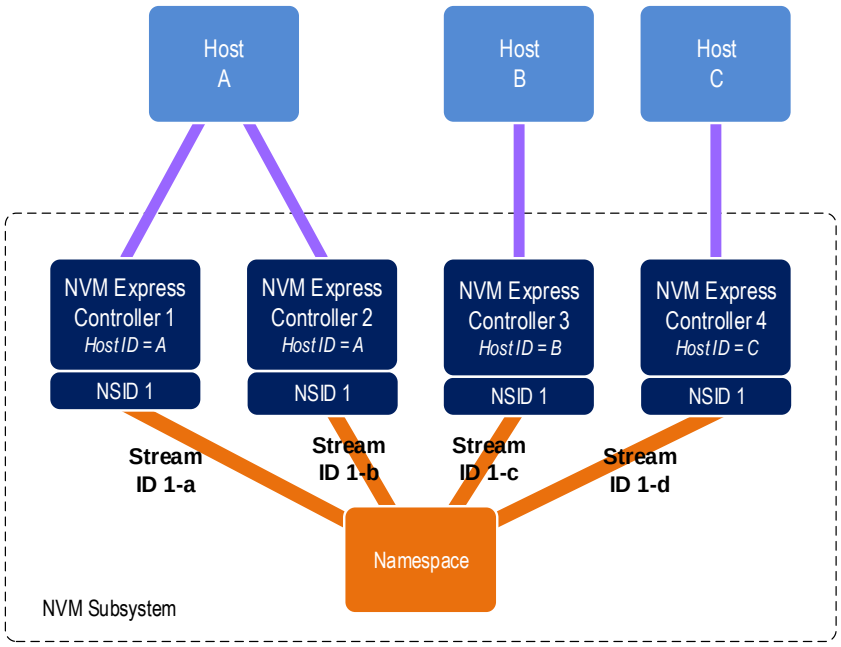
The host may access a namespace through multiple controllers in the NVM subsystem. The controllers in an NVM subsystem indicate in bit 0 of the NSSC field (refer to Figure 425) if a stream identifier is unique based on the Host Identifier (i.e., the same stream identifier used to access the same namespace by a host that has registered a different Host Identifier is referencing a different stream), or if a stream identifier may be used by multiple Host Identifiers (i.e., the same stream identifier used to access the same namespace by a host that has registered a different Host Identifier is referencing the same stream). All controllers in an NVM subsystem shall report the same value in the NSSC field.

If multiple controllers receive a registration of a Host Identifier (refer to section 5.27.1.16) that has the same non-zero value, then that value represents a single host that is accessing the namespace through those controllers and a stream identifier is used across those controllers to access the same stream on the namespace. If a Host Identifier has a unique non-zero value, then each value represents a unique host that is accessing the namespace and:

- if bit 0 of the NSSC field is cleared to '0', then the same stream identifier on controllers with different non-zero Host Identifiers does not have the same meaning for a particular namespace (i.e., the stream identifier is not used across controllers with different non-zero Host Identifiers to access the same stream on the namespace); and
- if bit 0 of the NSSC field is set to '1', then the same stream identifier on any controller with a non-zero Host Identifier has the same meaning for a particular namespace (i.e., the stream identifier is used across controllers to access the same stream on the namespace).

If a Host Identifier is cleared to 0h, then a unique host is accessing the namespace and the stream identifier does not have the same meaning for a particular namespace.

Figure 423: Example Multi-Stream and NSSC



In the example shown in Figure 423, if NSSC bit 0 is cleared to '0', then there are three streams as follows:

- Stream ID 1-a and Stream ID 1-b have the same meaning;
- Stream ID 1-c has a different meaning; and
- Stream ID 1-d has a different meaning.

In the example shown in Figure 423, if NSSC bit 0 is set to '1', then there is one stream as follows:

- Stream ID 1-a, Stream ID 1-b, Stream ID 1-c, and Stream ID 1-d have the same meaning.

The controller(s) recognized by the NVM subsystem as being associated with a specific host or hosts and attached to a specific namespace either:

- utilizes a number of stream resources allocated for exclusive use of that namespace as returned in response to an Allocate Resources operation; or
- utilizes resources from the NVM subsystem stream resources.

The value of Namespace Streams Allocated (NSA) indicates how many resources for individual stream identifiers have been allocated for exclusive use for the specified namespace by the associated controllers. This indicates the maximum number of stream identifiers that may be open at any given time in the specified namespace by the associated controllers. To request a different number of resources than are currently allocated for exclusive use by the associated controllers for a specific namespace, all currently allocated resources are first required to be released using the Release Resources operation. There is no mechanism to incrementally increase or decrease the number of allocated resources for a given namespace.

Streams are opened by the controller when the host issues a Write command that specifies a stream identifier that is not currently open. While a stream is open the controller maintains context for that stream (e.g., buffers for associated data). The host may determine the streams that are open using the Get Status operation.

For a namespace that has a non-zero value of Namespace Streams Allocated (NSA), if the host submits a Write command specifying a stream identifier not currently in use and stream resources are exhausted, then an arbitrary stream identifier for that namespace is released by the controller to free the stream resources associated with that stream identifier for the new stream. The host may ensure the number of open streams does not exceed the allocated stream resources for the namespace by explicitly releasing stream identifiers as necessary using the Release Identifier operation.

For a namespace that has zero namespace stream resources allocated, if the host submits an I/O command specifying a stream identifier not currently in use and:

- NVM subsystem streams available are exhausted, then an arbitrary stream identifier for an arbitrary namespace that is using NVM subsystem stream resources is released by the NVM subsystem to free the stream resources associated with that stream identifier for the new stream; or
- all NVM subsystem stream resources have been allocated for exclusive use for specific namespaces, then the Write command is treated as a normal Write command that does not specify a stream identifier.

The host determines parameters associated with stream resources using the Return Parameters operation. The host may get a list of open stream identifiers using the Get Status operation.

If the Streams Directive becomes disabled for use by a host within a namespace, then all stream resources and stream identifiers shall be released for that host for the affected namespace. If the host issues a Format NVM command, then all stream identifiers for all open streams for affected namespaces shall be released. If the host deletes a namespace, then all stream resources and all stream identifiers for that namespace shall be released. If the write protection state of a namespace changes such that the namespace becomes write protected (refer to section 8.12), then the controller shall release all stream resources and stream identifiers for that namespace.

Streams Directive defines the command specific status values specified in Figure 424.

Figure 424: Streams Directive – Command Specific Status Values

Value	Description
7Fh	**Stream Resource Allocation Failed:** The controller was not able to allocate stream resources for exclusive use of the specified namespace and no NVM subsystem stream resources are available.

8.7.3.1 Directive Receive

This section defines operations used with the Directive Receive command for the Streams Directive.

8.7.3.1.1 Return Parameters (Directive Operation 01h)

The Return Parameter operation returns a data structure that specifies the features and capabilities supported by the Streams Directive, including namespace specific values. The DSPEC field in command Dword 11 is not used for this operation. The data structure returned is defined in Figure 425. If an NSID value of FFFFFFFFh is specified, then the controller:

- returns the NVM subsystem specific values;
- may return any namespace specific values that are the same for all namespaces (e.g., SWS); and
- clears all other namespace specific fields to 0h.

Figure 425: Streams Directive – Return Parameters Data Structure

Bytes	Description
	NVM Subsystem Specific Fields
01:00	**Max Streams Limit (MSL):** This field indicates the maximum number of concurrently open streams that the NVM subsystem supports. This field returns the same value independent of specified namespace.
03:02	**NVM Subsystem Streams Available (NSSA)**: This field indicates the number of NVM subsystem stream resources available. These are the stream resources that are not allocated for the exclusive use by a host in any specific namespace. This field returns the same value independent of specified namespace.
05:04	**NVM Subsystem Streams Open (NSSO)**: This field indicates the number of open streams in the NVM subsystem that are not associated with a namespace for which resources were allocated using an Allocate Resources operation. This field returns the same value independent of specified namespace.

Figure 425: Streams Directive – Return Parameters Data Structure

Bytes	Description
06	**NVM Subsystem Stream Capability (NSSC):** This field indicates the stream capabilities of the NVM subsystem. Bits 7:1 are reserved. Bit 0 indicates whether stream identifiers may be shared by multiple Host Identifiers, or if a stream identifier is associated with a single Host Identifier. If bit 0 is cleared to '0', then the stream identifier is associated with a single non-zero Host Identifier. If bit 0 is set to '1', then the stream identifier may be associated with multiple non-zero Host Identifiers.
15:07	Reserved
Namespace Specific Fields	
19:16	**Stream Write Size (SWS):** This field indicates the alignment and size of the optimal stream write as a number for the specified namespace where the unit of granularity is specified by the applicable I/O Command Set. The size indicated should be less than or equal to Maximum Data Transfer Size (MDTS) that is specified in units of minimum memory page size. SWS may change if the namespace is reformatted with a different User Data Format. If the NSID value is set to FFFFFFFFh, then this field may be cleared to 0h if a single user data size cannot be indicated. Refer to the applicable I/O Command Set specification for how this field is utilized to optimize performance and endurance.
21:20	**Stream Granularity Size (SGS):** This field indicates the stream granularity size for the specified namespace in Stream Write Size (SWS) units. If the NSID value is set to FFFFFFFFh, then this field may be cleared to 0h. Refer to the applicable I/O Command Set specification for how this field is utilized to optimize performance and endurance.
Namespace and Host Identifier Specific Fields	
23:22	**Namespace Streams Allocated (NSA):** This field indicates the number of stream resources allocated for exclusive use of the specified namespace. If bit 0 of the NSSC field is cleared to '0', then those exclusive stream resources are shared by the controller processing the Return Parameters operation and by all other controllers that share the same non-zero Host Identifier, and are attached to the specified namespace. If bit 0 of the NSSC field is set to '1', then those exclusive stream resources are shared by all controllers that are associated with any non-zero Host Identifier and are attached to this namespace. If this value is non-zero, then the namespace may have up to NSA number of concurrently open streams. If this field is cleared to 0h, then no stream resources are currently allocated to this namespace and the namespace may have up to NSSA number of concurrently open streams.
25:24	**Namespace Streams Open (NSO)**: This field indicates the number of open streams in the specified namespace. If bit 0 of the NSSC field is cleared to '0', then this field indicates the number of streams that were opened by the controller processing the Return Parameters operation and by all other controllers that share the same non-zero Host Identifier, and are attached to this namespace. If bit 0 of the NSSC field is set to '1', then this field indicates the number of streams that were opened by the controller processing the Return Parameters operation and all other controllers that are associated with any non-zero Host Identifier and are attached to this namespace. NOTE: It is not possible for a host to retrieve the number of open streams using resources allocated to the specified namespace by other hosts.
31:26	Reserved

8.7.3.1.2 Get Status (Directive Operation 02h)

The Get Status operation returns information about the status of currently open streams for the specified namespace and the host issuing the Get Status operation. The DSPEC field in command Dword 11 is not used for this operation.

If NSSC bit 0 is cleared to '0', then the information returned describes only those resources for the specified namespace that are associated with hosts that are registered with the same non-zero Host Identifier value as the host issuing the Get Status operation. If NSSC bit 0 is set to '1', then the information returned describes the resources for the specified namespace that are associated with hosts that are registered with any non-zero Host Identifier.

If an NSID value of FFFFFFFFh is specified, then the controller shall return information about the status of currently open streams in the NVM subsystem that use resources which are not allocated for the exclusive use of a particular namespace. If a stream identifier value being returned is in use by different namespaces, then that stream identifier shall be returned only once.

Stream Identifier 1 (i.e., returned at offset 03:02) contains the value of the open stream of lowest numerical value. Each subsequent field contains the value of the next numerically greater stream identifier of an open stream.

The data structure returned is defined in Figure 426. All fields are specific to the namespace specified if the NSID value was not set to FFFFFFFFh.

Figure 426: Streams Directive – Get Status Data Structure

Bytes	Description
01:00	**Open Stream Count:** This field specifies the number of streams that are currently open.
03:02	**Stream Identifier 1:** This field specifies the stream identifier of the first (numerically lowest) open stream.
05:04	**Stream Identifier 2:** This field specifies the stream identifier of the second open stream.
…	…
131071: 131070	**Stream Identifier 65,535:** This field specifies the stream identifier of the 65,535th open stream.

8.7.3.1.3 Allocate Resources (Directive Operation 03h)

The Allocate Resources operation indicates the number of streams that the host requests for the exclusive use for the specified namespace. If bit 0 of the NSSC field is cleared to '0', then those resources are for the exclusive use of hosts that are registered with the same Host Identifier as the host that made the request. If bit 0 of the NSSC field is set to '1', then those resources are for the exclusive use of any host that is registered with any non-zero Host Identifier. The DSPEC field in command Dword 11 is not used for this operation. The operation returns the number of streams allocated in Dword 0 of the completion queue entry. The value allocated may be less than or equal to the number requested. The allocated resources shall be reflected in the Namespace Streams Allocated field of the Return Parameters data structure.

If the controller is unable to allocate any stream resources for the exclusive use for the specified namespace, then the controller shall:

- return a status value of Stream Resource Allocation Failed; or
- if NVM subsystem stream resources are available, then clear NSA to 0h in the completion queue entry to indicate that the host may use stream resources from the NVM subsystem for this namespace.

If the specified namespace already has stream resources allocated for the exclusive use of the host issuing the Allocate Resources operation, then the controller shall return a status code of Invalid Field in Command. To allocate additional streams resources, the host should release resources and request a complete set of resources.

No data transfer occurs.

Figure 427: Allocate Resources – Command Dword 12

Bits	Description
31:16	Reserved
15:00	**Namespace Streams Requested (NSR):** This field specifies the number of stream resources the host is requesting be allocated for exclusive use by the namespace specified.

Figure 428: Allocate Resources – Completion Queue Entry Dword 0

Bits	Description
31:16	Reserved
15:00	**Namespace Streams Allocated (NSA):** This field indicates the number of streams resources that have been allocated for exclusive use by the namespace specified. The allocated resources are available to all controllers associated with that host.

8.7.3.2 Directive Send

This section defines operations used with the Directive Send command for the Streams Directive.

8.7.3.2.1 Release Identifier (Directive Operation 01h)

The Release Identifier operation specifies that the stream identifier specified in the DSPEC field in command Dword 11 is no longer in use by the host. Specifically, if the host uses that stream identifier in a future operation, then that stream identifier is referring to a different stream. If the specified identifier does not correspond to an open stream for the specified namespace, then the Directive Send command should not fail as a result of the specified identifier. If there are stream resources allocated for the exclusive use of the specified namespace, then those exclusive stream resources remain allocated for this namespace and may be re-used in a subsequent write command. If there are no stream resources allocated for the exclusive use of the specified namespace, then the stream resources are returned to the NVM subsystem stream resources for future use by a namespace without exclusive allocated stream resources. If an NSID value of FFFFFFFFh is specified, then the controller shall abort the command with a status code of Invalid Field in Command.

No data transfer occurs.

8.7.3.2.2 Release Resources (Directive Operation 02h)

The Release Resources operation is used to release all streams resources allocated for the exclusive use of the namespace attached to all controllers:

- associated with the same non-zero Host Identifier of the controller that processed the operation if bit 0 of the NSSC field is cleared to ‘0’; and
- associated with any non-zero Host Identifier if bit 0 of the NSSC field is set to ‘1’.

On successful completion of this command, the exclusive allocated stream resources are released and the Namespace Streams Allocated (refer to Figure 425) field is cleared to 0h for the specified namespace. If this command is issued when no streams resources are allocated for the exclusive use of the namespace, then the Directive Send command shall take no action and shall not fail as a result of no allocated stream resources.

No data transfer occurs.

8.8 Doorbell Stride for Software Emulation

The doorbell stride, specified in CAP.DSTRD (refer to Figure 36), may be used to separate doorbells by a number of bytes in memory space. The doorbell stride is a number of bytes equal to (2 ^ (2 + CAP.DSTRD)). This is useful in software emulation of an NVM Express controller. In this case, a software thread is monitoring doorbell notifications. The software thread may be made more efficient by monitoring one doorbell per discrete cacheline or utilize the monitor/mwait CPU instructions. For hardware implementations of the NVM Express interface, the expected doorbell stride value is 0h.

8.9 Host Memory Buffer

The Host Memory Buffer (HMB) feature allows the controller to utilize an assigned portion of host memory exclusively. The use of the host memory resources is vendor specific. Host software may not be able to provide any or a limited amount of the host memory resources requested by the controller. The controller shall function properly without host memory resources. Refer to section 5.27.1.10.

The controller may indicate limitations for the minimum usable descriptor entry size and the maximum number of descriptor entries (refer to the HMMINDS and HMMAXD fields in the Identify Controller data structure, Figure 275). If the host does not create the Host Memory Buffer within the indicated limits, then the host memory allocated for use by the controller may not be fully utilized (e.g., descriptor entries beyond the maximum number of entries indicated may be ignored by the controller).

During initialization, host software may provide a descriptor list that describes a set of host memory address ranges for exclusive use by the controller. The host memory resources assigned are for the exclusive use of the controller (host software should not modify the ranges) until host software requests that the controller release the ranges and the controller completes the Set Features command. The controller is responsible for initializing the host memory resources. Host software should request that the controller release the assigned ranges prior to a shutdown event, a Runtime D3 event, or any other event that requires host software to reclaim the assigned ranges. After the controller acknowledges that the ranges are no longer in use, host software may reclaim the host memory resources. In the case of Runtime D3, host software should provide the host memory resources to the controller again and inform the controller that the ranges were in use prior to the RTD3 event and have not been modified.

The host memory resources are not persistent in the controller across a reset event. Host software should provide the previously allocated host memory resources to the controller after the reset completes. If host software is providing previously allocated host memory resources (with the same contents) to the controller, the Memory Return bit is set to '1' in the Set Features command.

The controller shall ensure that there is no data loss or data corruption in the event of a surprise removal while the Host Memory Buffer feature is being utilized.

8.10 Host Operation with Asymmetric Namespace Access Reporting (Informative)

8.10.1 Host ANA Normal Operation

The host determines if ANA is supported by examining bit 3 in the CMIC field in the Identify Controller data structure (refer to Figure 275). The NSID or Identifier (refer to section 4.3) is used to determine when multiple paths to the same namespace are available. The host examines the ANA log page (refer to section 5.16.1.13) for each controller to determine the ANA state of each group of namespaces attached to that controller.

To send a command to a namespace, the host should select a controller that reports the ANA Optimized State (refer to section 8.1.3.1) and send the command to that controller. If more than one controller that reports the ANA Optimized state for a namespace are found, then the host may use all of those controllers to send commands.

If there are no controllers that report the ANA Optimized state for a namespace, then the host should select a controller that reports ANA Non-Optimized State (refer to section 8.1.3.2) for that namespace and send the command to that controller. If more than one controller that reports ANA Non-Optimized state for a namespace are found, then the host may use all of those controllers to send commands.

If multiple controllers are being used, then the algorithm for determining which controller to use next is outside the scope of this specification (e.g., the host may select a simple round robin algorithm, a queue depth weighted algorithm, a transfer length weighted algorithm, or any other algorithm).

If there are no controllers that report the ANA Optimized state for a namespace and there are no controllers that report the ANA Non-Optimized state for that namespace, then the host should examine controllers that report the ANA Inaccessible state as described in section 8.10.2.

8.10.2 Host ANA Inaccessible Operation

If the ANA log page reports an ANA state of ANA Inaccessible State for an ANA Group or a command returns a status code of Asymmetric Access Inaccessible, then the host should:

- not use that controller to send commands to any namespace in that ANA Group; and
- select a different controller for sending commands to all namespaces in that ANA Group.

If there are no controllers that report the ANA Optimized state for a namespace and there are no controllers that report the ANA Non-Optimized state, then a transition may be occurring that also impacts controllers that are reporting the ANA Inaccessible state. As a result, the host should use the methods described for Host ANA Transition operation (refer to section 8.10.5) to determine if the controller reporting ANA Inaccessible state transitions during the ANATT time interval to an ANA state that enables commands to be processed by that controller.

8.10.3 Host ANA Persistent Loss Operation

If the ANA log page reports an ANA state of ANA Persistent Loss State for an ANA Group or a command returns a status code of Asymmetric Access Persistent Loss, then the host should not use that controller to send commands to any namespace in that ANA Group, and select a different controller for sending commands to any namespace in that ANA Group. If the controller supports the Namespace Management capability (refer to section 8.11), then the namespaces in an ANA Group reporting this state should be detached.

8.10.4 Host ANA Change Notice Operation

If the ANA log page reports an ANA state of ANA Change State for an ANA Group or a command returns a status code of Asymmetric Access Transition, then the host should temporarily not use that controller to send commands to any namespace in that ANA Group. If only controllers reporting ANA Inaccessible State are available, then the host should follow these procedures to determine which controller to use. To use a controller, the host may:

a) if Asymmetric Namespace Access Change Notices are enabled (refer to section 5.27.1.8) on the controller, wait for an Asymmetric Namespace Access Change Notice from that controller. Upon receipt of that notice, the host should examine the ANA log page to determine the new ANA state and resume sending commands based on the new ANA state. Such notice should occur within the ANATT time (refer to Figure 275); or
b) delay and retry the command during the ANATT time interval. The host should not immediately retry, but rather, divide the ANATT time into equal intervals for command retry purposes (e.g., if ANATT is 30, perform 3 retries at 10 s intervals, or 10 retries at 3 s intervals). During or upon completion of the ANATT time interval, the new ANA state of the ANA Group should be known (e.g., one of the command retries returned a different status that indicates completion of the transition to a new ANA state). If the retried command did not complete without error, the ANA log page should be examined on each controller that provides access to the namespace and the host should resume sending commands based on the new ANA state.

If the ANATT time interval expires, then the host should use a different controller for sending commands to the namespaces in that ANA Group. The ANATT interval reported by the controller should prevent this type of timer expiration from occurring.

8.10.5 Host ANA Transition Operation

Receipt of an Asymmetric Namespace Access Change Notice from a controller may indicate:

a) that the ANA state reported in one or more ANA Group Descriptors has changed;
b) a new NSID has been added to one or more of the ANA Group Descriptors;
c) an NSID has been removed from one or more of the ANA Group Descriptors; and/or
d) the NSID of a namespace has moved from one ANA Group Descriptor to a different ANA Group Descriptor (i.e., the ANAGRPID field in the Identify Namespace data structure for that namespace has changed), if bit 6 in the ANACAP field is cleared to ‘0’ in the Identify Controller data structure (refer to Figure 275).

As a result of receiving an Asymmetric Namespace Access Change Notice, the host should read the ANA log page (refer to section 5.16.1.13) to check for each of those possible changes.

8.10.6 All Paths Down Condition

An all paths down condition occurs when there are no paths available on the host to access the namespaces in an ANA Group (i.e., the NVM media). To determine whether an all paths down condition has occurred, the host may examine the ANA log page on each controller that provides access to the namespaces in a particular ANA Group. All paths that are not in the ANA Persistent Loss state should be checked. If no paths to the namespaces in that ANA Group become available (i.e., transition to the ANA Optimized state or the ANA Non-Optimized state) for the duration of an ANATT time interval, then an all paths down condition has occurred for the namespaces in that ANA Group.

8.11 Namespace Management

The Namespace Management capability consists of the Namespace Management command (refer to section 5.23) and the Namespace Attachment command (refer to section 5.22). The Namespace Management command is used to create a namespace or delete a namespace. The Namespace Attachment command is used to attach and detach controllers from a namespace. The Namespace Management capability is intended for use during manufacturing or by a system administrator.

If the Namespace Management capability is supported, then the controller:

a) shall support the Namespace Management command and the Namespace Attachment command;
b) shall set bit 3 to ‘1’ in the OACS field (refer to Figure 275);
c) should support the Namespace Attribute Changed asynchronous event (refer to Figure 147 and section 5.27.1.8); and
d) may support Namespace Granularity (refer to the NVM Command Set Specification).

If a namespace is detached from a controller, then the NSID that referred to that namespace becomes an inactive NSID (refer to section 3.2.1.4) on that controller. If a namespace is deleted from the NVM subsystem, then the NSID that referred to that namespace becomes an unallocated NSID (refer to section 3.2.1.3) in the NVM subsystem. Previously submitted but uncompleted or subsequently submitted commands to the affected NSID are handled by the controller as if they were issued to an inactive NSID (refer to Figure 87).

The size of a namespace is based on the size requested in a create operation, the format of the namespace, and any characteristics (e.g., endurance). The controller determines the NVM capacity allocated for that namespace. Namespaces may be created with different usage characteristics (e.g., endurance) that utilize differing amounts of NVM capacity. Namespace characteristics and the mapping of these characteristics to NVM capacity usage are outside the scope of this specification.

Reporting of capacity information for the NVM subsystem, Domain, Endurance Group, and NVM Set are described in section 3.8. For each namespace, the NVM Set and the Endurance Group that contain the namespace are reported in the Identify Namespace data structure. The NVM Set to be used for a namespace is based on the value in the NVM Set Identifier field in a create operation. If the NVM Set Identifier field is cleared to 0h in a create operation, then the controller shall choose the NVM Set from which to allocate capacity to create the namespace.

If the NVM Set Identifier field and the Endurance Group Identifier field are both cleared to 0h in a create operation, then the controller shall choose the Endurance Group and the NVM Set from which to allocate capacity to create the namespace.

If the NVM Set Identifier field is cleared to 0h and the Endurance Group Identifier field is set to a non-zero value in a create operation, then the controller shall choose the NVM Set in the specified Endurance Group from which to allocate capacity to create the namespace.

If the NVM Set Identifier field is set to a non-zero value and the Endurance Group Identifier field is cleared to 0h in a create operation, then the controller shall abort the command with a status code of Invalid Field in Command.

If the NVM Set Identifier field and the Endurance Group Identifier field are both set to non-zero values in a create operation and the specified NVM Set exists in the specified Endurance Group, then the controller shall allocate capacity for the created namespace from the specified NVM Set.

If the NVM Set Identifier field and the Endurance Group Identifier field are both set to non-zero values in a create operation and the specified NVM Set does not exist in the specified Endurance Group, then the controller shall abort the command with a status code of Invalid Field in Command.

For each namespace, the NVM capacity used for that namespace is reported in the Identify Namespace data structure (refer to the applicable I/O Command Set specification). The controller may allocate NVM capacity in units such that the requested size for a namespace may be rounded up to the next unit boundary. The units in which NVM capacity is allocated are reported in the Namespace Granularity List (refer to the NVM Command Set Specification), if supported. For example when using the NVM Command Set, if host software requests a namespace of 32 logical blocks with a logical block size of 4 KiB for a total size of 128 KiB and the allocation unit for the implementation is 1 MiB, then the NVM capacity consumed may be rounded up to 1 MiB. The NVM capacity fields may not correspond to the logical block size multiplied by the total number of logical blocks.

The method of allocating ANA Group identifiers is outside the scope of this specification. If the ANA Group Identifier (refer to Figure 280 and the Identify Namespace data structure in the NVM Command Set Specification) is cleared to 0h, then the controller shall determine the ANAGRPID that is assigned to that namespace.

To create a namespace, host software performs the following actions:

1. Host software requests the Identify Namespace data structure that specifies common namespace capabilities (i.e., using an Identify command with the NSID field set to FFFFFFFFh and the CNS field cleared to 0h);
2. If the controller supports reporting of I/O Command Set specific Namespace Management content (refer to the Namespace Management section in the applicable I/O Command Set specification), host software optionally requests that information (e.g. Namespace Granularity).
3. Host software determines available capacity (refer to section 3.8);
4. Host software creates the data structure defined in Figure 300 (e.g., taking into account the common namespace capabilities, available capacity);
5. Host software issues the Namespace Management command specifying the Create operation and the data structure. On successful completion of the command, the Namespace Identifier of the new namespace is returned in Dword 0 of the completion queue entry. At this point, the new namespace is not attached to any controller; and
6. Host software requests the Identify Namespace data structures for the new namespace to determine all attributes of the namespace.

To attach a namespace, host software performs the following actions:

1. Host software issues the Namespace Attachment command specifying the Controller Attach operation to attach the specified namespace to one or more controllers; and
2. If Namespace Attribute Notices are enabled, the controller(s) newly attached to the namespace report a Namespace Attribute Changed asynchronous event to the host.

To detach a namespace, host software performs the following actions:

1. Host software issues the Namespace Attachment command specifying the Controller Detach operation to detach the specified namespace from one or more controllers; and
2. If Namespace Attribute Notices are enabled, the controllers that were detached from the namespace report a Namespace Attribute Changed asynchronous event to the host.

To delete a namespace, host software performs the following actions:

1. Host software should detach the namespace from all controllers;
2. Host software issues the Namespace Management command specifying the Delete operation for the specified namespace. On successful completion of the command, the namespace has been deleted; and

3. If Namespace Attribute Notices are enabled, any controller(s) not processing the Namespace Management command that was attached to the namespace reports a Namespace Attribute Changed asynchronous event to the host.

8.12 Namespace Write Protection

Namespace Write Protection is an optional configurable controller capability that enables the host to control the write protection state of a namespace or to determine the write protection state of a namespace. Support for this capability is reported in the Namespace Write Protection Capabilities (NWPC) field in the Identify Controller data structure (refer to Figure 275).

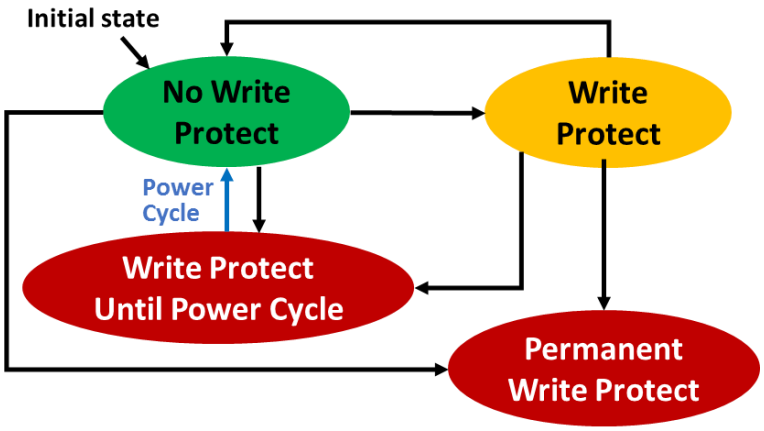
Figure 429 defines the write protection states that may be supported for a namespace. All states persist across power cycles and Controller Level Resets (refer to section 3.7.2) except Write Protect Until Power Cycle state, which transitions to the No Write Protect state on the occurrence of a power cycle.

Figure 429: Namespace Write Protection State Definitions

State	Definition	Persistent Across	
		Power Cycles	Controller Level Resets
No Write Protect	The namespace is not write protected.	Yes	Yes
Write Protect	The namespace is write protected.	Yes	Yes
Write Protect Until Power Cycle	The namespace is write protected until the next power cycle.	No	Yes
Permanent Write Protect	The namespace is permanently write protected.	Yes	Yes

Figure 430 defines the transition between write protection states. All state transitions are based on Set Features commands unless otherwise specified. The initial state of a namespace at the time of its creation is the No Write Protect state.

Figure 430: Namespace Write Protection State Machine Model



The Write Protect Until Power Cycle and Permanent Write Protect states are subject to the Namespace Write Protection Authentication Control mechanism, which determines whether the controller processes or aborts Set Features commands which cause a transition into either of these two states (refer to section 8.18).

The results of using Namespace Write Protection in combination with an external write protection system (e.g., TCG Storage Interface Interactions Specification) are outside the scope of this specification.

8.12.1 Namespace Write Protection – Theory of Operation

If Namespace Write Protection is supported by the controller, then the controller shall:

- Indicate the level of support for Namespace Write Protection capabilities in the Namespace Write Protection Capabilities (NWPC) field in the Identify Controller data structure; and
- Support the Namespace Write Protection Config Feature (refer to section 5.27.1.28).

If the Write Protect Until Power Cycle or the Permanent Write Protect states are supported by the controller, then the controller shall support the Namespace Write Protection Authentication Control field in the RPMB Device Configuration Block data structure (refer to section 8.18).

The controller shall not set the Critical Warning field, bit 3 (refer to Figure 207) to '1' if the read-only condition on the media is a result of a change in the namespace write protection state as defined by the Namespace Write Protection State Machine (refer to Figure 430), or due to any autonomous namespace write protection state transitions (e.g., power cycle). Host software may check the current namespace write protection state of a namespace using the Get Features command with the Namespace Write Protection Config Feature Identifier.

If any controller in the NVM subsystem supports Namespace Write Protection, then the write protection state of a namespace shall be enforced by any controller to which that namespace is attached.

8.12.1.1 Namespace Write Protection – Command Interactions

Unless otherwise noted, the commands listed in Figure 431 are processed normally when specifying an NSID for a namespace that is write protected.

Figure 431: Commands Allowed when Specifying a Write Protected NSID

Admin Command Set	NVM Command Set
Device Self–test	Compare
Directive Send [1]	Dataset Management [1]
Directive Receive [3]	Read
Get Features	Reservation Register
Get Log Page	Reservation Report
Identify	Reservation Acquire
Namespace Attachment	Reservation Release
Security Receive [1]	Vendor Specific [1]
Security Send [1]	Flush [2]
Set Features [1]	Verify
Vendor Specific [1]	

Notes:
1. The controller shall fail commands if the specified action attempts to modify the non-volatile storage medium of the specified namespace.
2. A Flush command shall complete successfully with no effect. All volatile write cache data and metadata associated with the specified namespace is written to non-volatile storage medium as part of transitioning to the write protected state (refer to section 5.27.1.28).
3. A Directive Receive command which attempts to allocate streams resources shall be aborted with a status code of Namespace is Write Protected.

Commands not listed in Figure 431, and which meet the following conditions, shall be aborted with a status code of Namespace Is Write Protected (refer to Figure 94):

a) Commands that specify an NSID for a namespace that is write protected;
b) Commands that specify an NSID for a namespace that is not write protected and the execution of which would modify another namespace that is write protected (e.g., a Format NVM command); and
c) Commands that do not specify an NSID, and the execution of which would modify a namespace that is write protected (e.g., Sanitize command).

8.13 NVMe over Fabrics Secure Channel and In-band Authentication

NVMe over Fabrics supports both fabric secure channel (that includes authentication) and NVMe in-band authentication. Fabric authentication is part of establishing a fabric secure channel via an NVMe Transport specific protocol that provides authentication, encryption, and integrity checking (e.g., IPsec; refer to RFC 4301 or TLS; refer to RFC 8446). NVMe in-band authentication is performed immediately after a Connect command (refer to section 6.3) succeeds using the Authentication Send and Authentication Receive commands (refer to section 6.1 and section 6.2) to tunnel authentication protocol commands between the host and the controller.

Enrollment of the host and controller in an authentication mechanism, including provisioning of authentication credentials to the host and controller, is outside the scope of this specification.

If both fabric secure channel and NVMe in-band authentication are used, the identities for these two instances of authentication may differ for the same NVMe Transport connection. For example, if an iWARP NVMe Transport is used with IPsec as the fabric secure channel technology, the IPsec identities for authentication are associated with the IP network (e.g., DNS host name or IP address), whereas NVMe in-band authentication uses NVMe identities (i.e., Host NQNs). The NVMe Transport binding specification may provide further guidance and requirements on the relationship between these two identities, but determination of which NVMe Transport identities are authorized to be used with which NVMe identities is part of the security policy for the deployed NVM subsystem.

8.13.1 Fabric Secure Channel

The Transport Requirements field in the Fabrics Discovery Log Page Entry (refer to Figure 264) indicates whether a fabric secure channel shall be used for an NVMe Transport connection to an NVM subsystem. The secure channel mechanism is specific to the type of fabric.

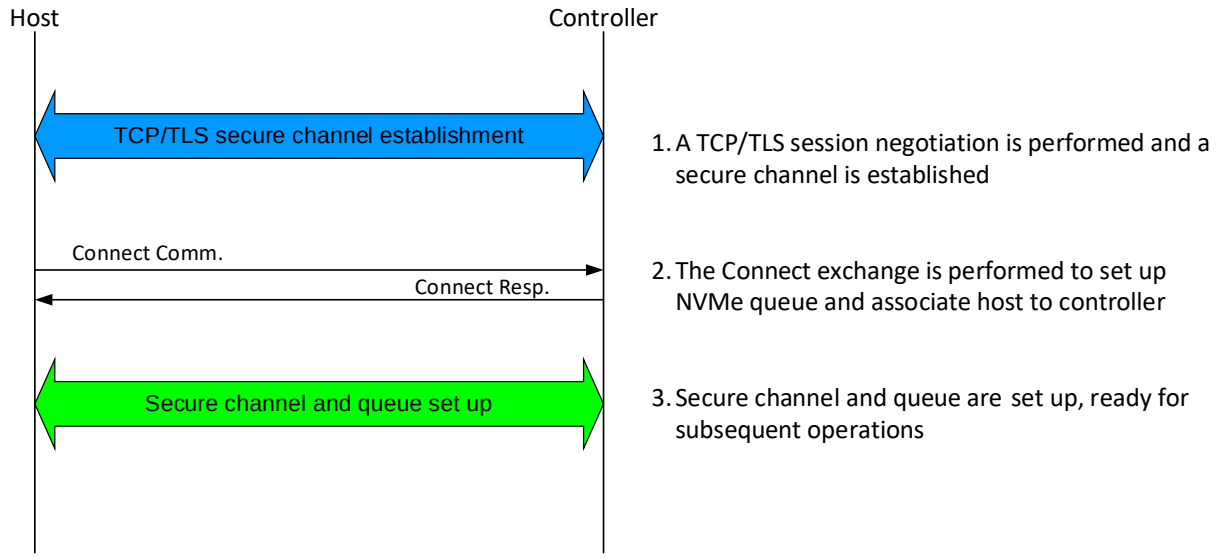
If establishment of a secure channel fails or a secure channel is not established when required by the controller, the resulting errors are fabric-specific and may not be reported to the NVMe layer on the host. Such errors may result in the controller being inaccessible to the host via the NVMe Transport connection on which the failure to establish a fabric secure channel occurred.

An NVM subsystem that requires use of a fabric secure channel (i.e., as indicated by the TREQ field in the associated Discovery Log Page Entry) shall not allow capsules to be transferred until a secure channel has been established for the NVMe Transport connection.

All Discovery Log Page Entries for an NVM subsystem should report the same value of TREQ to each host. Discovery Log Page Entries for an NVM subsystem may report different values of TREQ to different hosts.

Figure 432 shows an example of secure channel establishment using TLS.

Figure 432: Example of TLS secure channel establishment



8.13.2 NVMe In-band Authentication

The Authentication and Security Requirements (AUTHREQ) field in the Connect response capsule (refer to Figure 382) indicates whether NVMe in-band authentication is required.

If one or more of the bits in the AUTHREQ field are set to '1', then the controller requires that the host authenticate on that queue in order to proceed with Fabrics, Admin, and I/O commands. Authentication success is defined by the specific security protocol that is used for authentication. If any command other than Connect, Authentication Send, or Authentication Receive is received prior to authentication success, then the controller shall abort the command with Authentication Required status.

If all bits in the AUTHREQ field are cleared to '0', then the controller does not require the host to authenticate, and the NVM subsystem shall not abort any command with a status code value of Authentication Required.

If NVMe in-band authentication succeeds, then any supported commands for the associated queue type may be processed.

The host may initiate a subsequent authentication transaction at any time for reauthentication purposes. Initiating reauthentication shall not invalidate a prior authentication. If the reauthentication transaction concludes with the controller sending an AUTH_Failure1 message (refer to section 8.13.4.2), then the controller shall terminate all commands with a status code of Operation Denied and disconnect the NVMe over Fabrics connection. If the reauthentication transaction concludes with the host sending an AUTH_Failure2 message, then the host shall disconnect the NVMe over Fabrics connection.

The state of an in-progress authentication transaction is soft-state. If the subsequent command in an authentication transaction is not received by the controller within a timeout equal to:

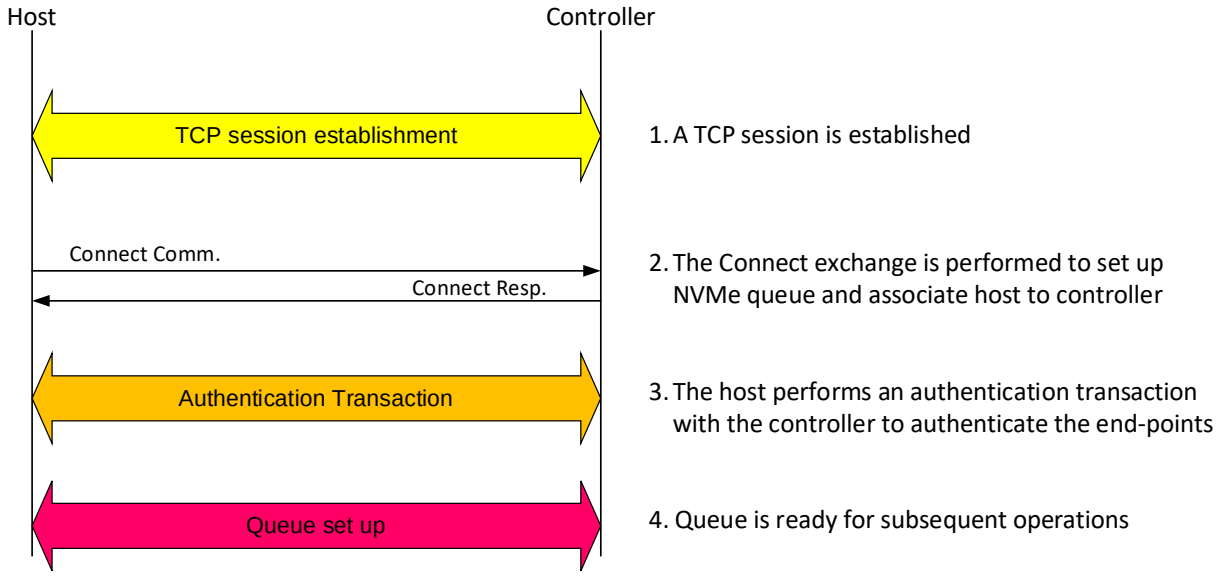
- the Keep Alive Timeout value (refer to Figure 380), if the Keep Alive Timer is enabled; or
- the default Keep Alive Timeout value (i.e., two minutes), if the Keep Alive Timer is disabled;

then the authentication transaction has timed out and the controller should discard the authentication transaction state (including the T_ID value, refer to section 8.13.4.1).

For an initial authentication, an authentication transaction timeout should be treated as an authentication failure with termination of the transport connection. For reauthentication, an authentication transaction timeout should not be treated as an authentication failure. Authentication commands used to continue that transaction after an authentication transaction timeout should be aborted with a status code of Command Sequence Error.

Figure 433 shows an example of authentication transaction for NVMe/TCP.

Figure 433: Example of authentication transaction for NVMe/TCP



8.13.2.1 NVMe In-band Authentication Protocol-Specific Requirements

Authentication requirements for security commands are based on the security protocol indicated by the SECP field in the command.

The authentication protocols defined by this specification use the security protocol identifier E9h (assigned to NVMe by SPC-5, a SCSI standard). The messages of the defined authentication protocols are self-identifying, therefore the SPSP0 field and the SPSP1 field of the Authentication Send and Authentication Receive commands shall be set to 01h. Authentication messages are mapped to NVMe over Fabrics command and response pairs. The mapping of authentication messages to the Authentication Send command is shown in Figure 434.

Figure 434: Mapping of authentication messages to the Authentication Send command

Field[1]	Value
SPSP0	01h
SPSP1	01h
SECP	E9h
TL	Specifies the amount of data to transfer in bytes
Notes: 1. Refer to section 6.2.	

The mapping of authentication messages to the Authentication Receive command is shown in Figure 435. Security processing requirements associated with the Authentication Receive command (e.g., delays in third-party authentication verification) may result in delays in controller completion of an Authentication Receive command. The host should consider these possible delays associated with the Authentication Receive command.

Figure 435: Mapping of authentication messages to the Authentication Receive command

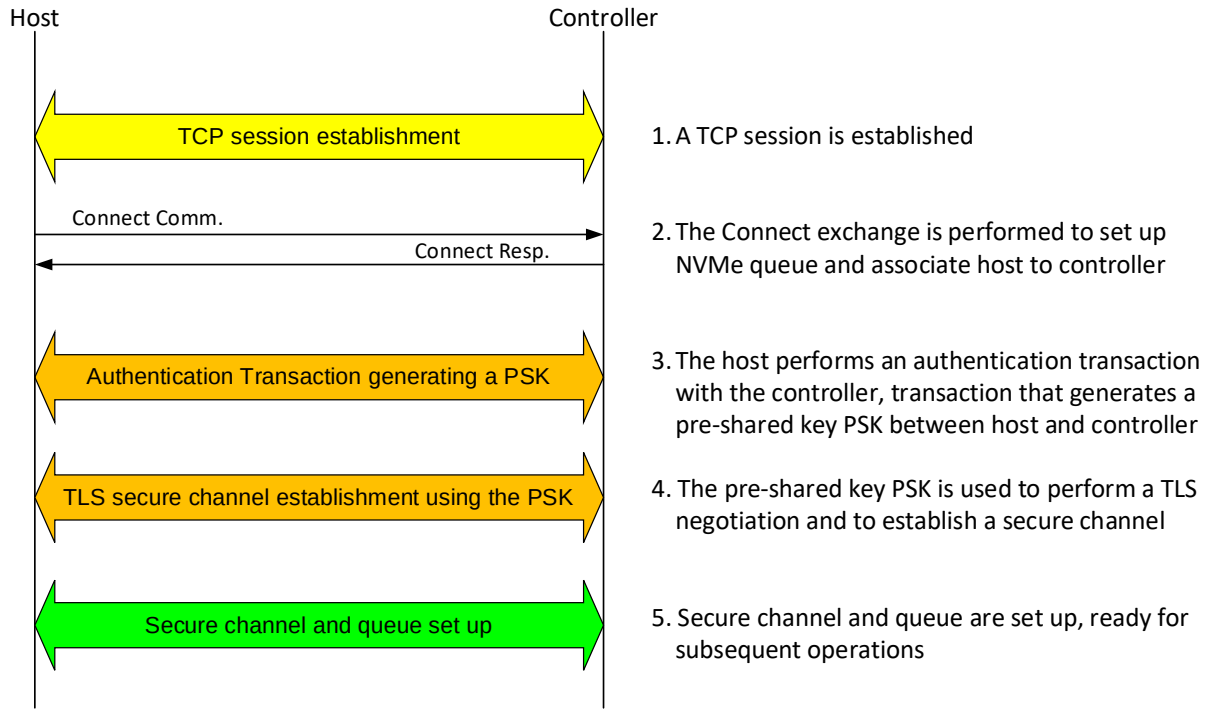
Field[1]	Value
SPSP0	01h

Field[1]	Value
SPSP1	01h
SECP	E9h
AL	Specifies the amount of data to transfer in bytes[2]
Notes: 1. Refer to section 6.1. 2. The size of the largest authentication message that could be received.	

8.13.3 NVMe In-band Authentication Followed by Secure Channel Establishment

It is possible to leverage an authentication transaction to generate shared key material to use as pre-shared key (PSK) to establish a secure channel (e.g., with IPsec or TLS). In this case, the PSK generated to set up a secure channel on the Admin Queue may be reused to set up additional secure channels on the I/O queues. Figure 436 shows an example of this possibility for TLS.

Figure 436: Example of TLS secure channel concatenated to an authentication transaction



8.13.4 Common Authentication Messages

8.13.4.1 AUTH_Negotiate Message

The AUTH_Negotiate message is sent from the host to the controller and is used to indicate the authentication protocols the host is able to use in this authentication transaction and which secure channel protocol, if any, to concatenate to this authentication transaction. The AUTH_Negotiate message format is shown in Figure 437.

Figure 437: AUTH_Negotiate message format

Bytes	Description
0	**AUTH_TYPE:** 00h (i.e., common messages)
1	**AUTH_ID:** 00h (i.e., AUTH_Negotiate)
3:2	Reserved
5:4	**T_ID:** 16-bit transaction identifier

Bytes	Description
6	**SC_C:** Secure channel concatenation
7	**NAPD:** Number of authentication protocol descriptors
71:8	Authentication protocol descriptor #1
135:72	Authentication protocol descriptor #2
…	
NAPD*64+7:(NAPD-1)*64+8	Authentication protocol descriptor #NAPD

The SC_C field determines if a secure channel concatenation to the authentication transaction is requested and with which secure channel protocol, as shown in Figure 438.

Figure 438: Secure channel protocol identifiers

Value	Description	Transport Applicability
00h	No secure channel concatenation	n/a
01h	Secure channel concatenation with TLS (refer to section 8.13.5.9)	TCP
All other values	Reserved	

The AUTH_Negotiate message is structured as a list of 64-byte authentication protocol descriptors to enable extensibility to define additional authentication protocols. Currently only one authentication protocol is defined (i.e., DH-HMAC-CHAP), therefore the AUTH_Negotiate message carries only one authentication protocol descriptor (i.e., NAPD=1). Implementations should support more than one descriptor to enable protocol extensibility. The first byte of an authentication protocol descriptor identifies the specific authentication protocol, as shown in Figure 439.

Figure 439: Authentication protocol identifiers

Value	Description
01h	DH-HMAC-CHAP (refer to section 8.13.5)
All other values	Reserved

Upon receiving an AUTH_Negotiate message, if the SC_C value indicated by the host does not satisfy the security requirements of the controller (e.g., the host did not request secure channel concatenation, but the controller's security policy requires secure channel concatenation), then the controller shall:

- reply to the AUTH_Negotiate message with an AUTH_Failure1 message having reason code 'Authentication failure' and reason code explanation 'Secure channel concatenation mismatch'; and
- disconnect the NVMe over Fabrics connection upon transmitting the AUTH_Failure1 message.

Upon receiving an AUTH_Negotiate message, if the protocol descriptors proposed by the host do not satisfy the security requirements of the controller, then the controller shall:

- reply to the AUTH_Negotiate message with an AUTH_Failure1 message having reason code 'Authentication failure' and reason code explanation 'Authentication protocol not usable'; and
- disconnect the NVMe over Fabrics connection upon transmitting the AUTH_Failure1 message.

8.13.4.2 AUTH_Failure Messages

The AUTH_Failure1 message is sent from the controller to the host, the AUTH_Failure2 message is sent from the host to the controller. The format of the AUTH_Failure1 message and of the AUTH_Failure2 message is shown in Figure 440.

Figure 440: AUTH_Failure1 and AUTH_Failure2 message format

Bytes	Description
0	**AUTH_TYPE:** 00h (i.e., common messages)
1	**AUTH_ID:** F0h (i.e., AUTH_Failure2) F1h (i.e., AUTH_Failure1)
3:2	Reserved
5:4	**T_ID:** 16-bit transaction identifier
6	Reason code
7	Reason code explanation

The AUTH_Failure reason codes are listed in Figure 441.

Figure 441: AUTH_Failure reason codes

Value	Description
01h	**Authentication failure:** The authentication transaction failed
All other values	Reserved

The AUTH_Failure reason code explanations are listed in Figure 442.

Figure 442: AUTH_Failure reason code explanations

Value	Description
01h	**Authentication failed:** Authentication of the involved host or NVM subsystem failed.
02h	**Authentication protocol not usable:** The protocol descriptors proposed by the host do not satisfy the security requirements of the controller (refer to section 8.13.4.1).
03h	**Secure channel concatenation mismatch:** The SC_C value indicated by the host does not satisfy the security requirements of the controller (refer to section 8.13.4.1).
04h	**Hash function not usable:** The HashIDList proposed by the host does not satisfy the security requirements of the controller (refer to section 8.13.5.2).
05h	**DH group not usable:** The DHgIDList proposed by the host does not satisfy the security requirements of the controller (refer to section 8.13.5.2).
06h	**Incorrect payload:** The payload of the received message is not correct.
07h	**Incorrect protocol message:** The received message is not the expected next message in the authentication protocol sequence.
All other values	Reserved

8.13.4.3 Mapping of Common Authentication Messages to Authentication Commands

The AUTH_Negotiate message and the AUTH_Failure2 message are sent from the host to the controller, therefore they are mapped to the Authentication Send command. The AUTH_Failure1 message is sent from the controller to the host, therefore it is mapped to the Authentication Receive command.

8.13.5 DH-HMAC-CHAP Protocol

8.13.5.1 Protocol Operations

DH-HMAC-CHAP is a key based Authentication and key management protocol that uses the Challenge Handshake Authentication Protocol (CHAP, refer to RFC 1994) enhanced to use the Hashed Message Authentication Code (HMAC) mechanism (refer to RFC 2104) with stronger hash functions and augmented with an optional Diffie-Hellman (DH) exchange (refer to RFC 2631, clause 2.2.1). DH-HMAC-CHAP provides bidirectional or unidirectional Authentication between a host and a controller.

The Diffie-Hellman part of the protocol is optional. When the Diffie-Hellman part of the protocol is not used, DH-HMAC-CHAP is referred to as HMAC-CHAP. If insufficiently random keys are used (refer to section 8.13.5.7), HMAC-CHAP potentially allows a passive eavesdropper to discover the key through an off-line dictionary attack, so its usage should be minimized. DH-HMAC-CHAP provides strong protection from

passive eavesdroppers. However, an active attacker could reduce the operation of this protocol so that only HMAC-CHAP is used, and as a result gain sufficient information to mount an off-line dictionary attack on the HMAC-CHAP key.

An implementation that supports DH-HMAC-CHAP authentication shall support DH-HMAC-CHAP with a NULL DH exchange. All implementations of DH-HMAC-CHAP shall be configurable to require a DH exchange (i.e., to not use HMAC-CHAP).

In order to authenticate with the DH-HMAC-CHAP protocol, each host and NVM subsystem shall be provided with a DH-HMAC-CHAP key that is associated with the entity's NQN. Two entities may impersonate one another if they have the same key, therefore when the assigned keys are not different for each entity there is a security vulnerability (refer to section 8.13.5.7).

To authenticate another entity, an entity is required to either:

a) know the key associated with the entity to be authenticated; or
b) rely on a third party that knows the key to verify the authentication.

An example of a DH-HMAC-CHAP authentication transaction is shown in Figure 443, with the notation shown in Figure 444. The DH-HMAC-CHAP_Success2 message that is shown as a dashed line is used only for bidirectional authentication.

Figure 443: Example of DH-HMAC-CHAP authentication transaction

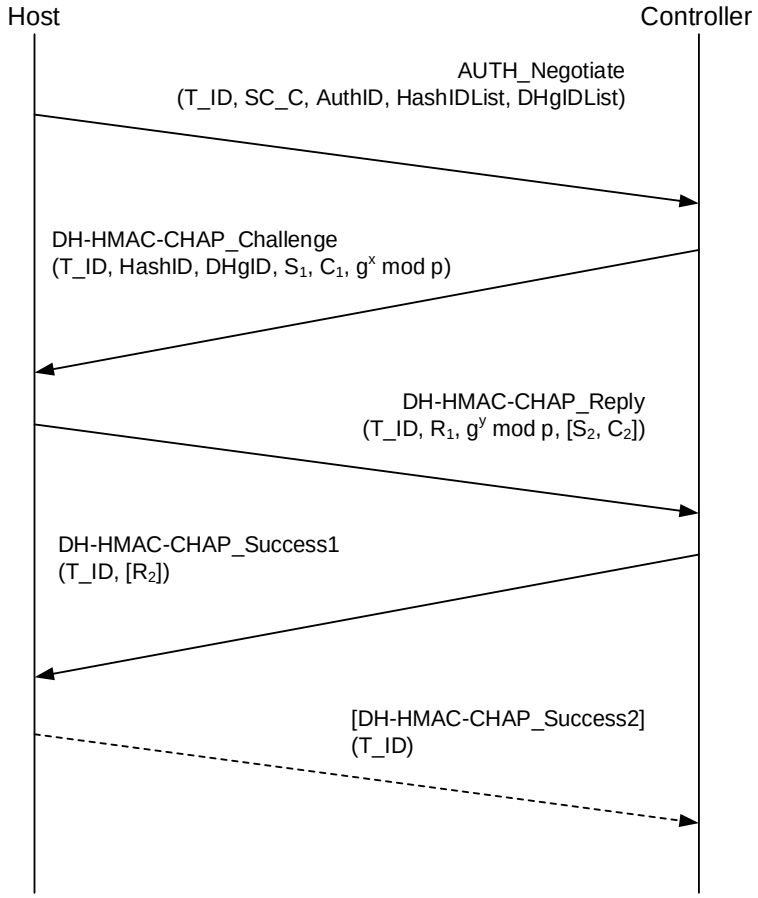


Figure 444: Mathematical notations for DH-HMAC-CHAP

Symbols	Description
NQN_c, NQN_h	NQN of the NVM subsystem that contains the controller and NQN of the host
K_c, K_h	DH-HMAC-CHAP key of the NVM subsystem that contains the controller and DH-HMAC-CHAP key of the host
p, g	Modulus (p) and generator (g) of the chosen DH group (refer to Figure 447)
x, y	Random numbers used as exponents in a DH exchange
C_1, C_2	Random challenge values
C_{a1}, C_{a2}	Augmented challenge values
S_1, S_2	32-bit sequence numbers
R_1, R_2	Reply values
T_ID	Authentication transaction identifier
SC_C	Secure channel concatenation indication
H( )	One-way hash function (refer to Figure 446)
HMAC(K, Str)	HMAC function (refer to RFC 2104) with key K on string Str using hash function H( )
||	Concatenation operation
K_S	Session key

When used with a non-NULL DH exchange, the DH-HMAC-CHAP protocol is able to generate a session key K_S to be used to establish a TLS session between host and controller (refer to section 8.13.5.9).

For an NVM subsystem, the controller is the entity running the protocol, using the identity and credentials of the NVM subsystem. The DH-HMAC-CHAP protocol proceeds in the following order:

1) The authentication transaction shall begin with the host sending the common AUTH_Negotiate message to negotiate the authentication protocol to use and its associated parameters (refer to section 8.13.4.1). The AUTH_Negotiate message carries the transaction identifier (T_ID) for the entire authentication transaction and the list of authentication protocol descriptors for the authentication protocols that may be used in this authentication transaction. For DH-HMAC-CHAP, the authentication protocol descriptor includes the list of hash functions (HashIDList) and Diffie-Hellman group identifiers (DHgIDList) that may be used in this authentication protocol transaction.

2) If the parameters of the received DH-HMAC-CHAP protocol descriptor are compatible with the controller's policies, then the controller shall reply with a DH-HMAC-CHAP_Challenge message (refer to section 8.13.5.3) carrying the same transaction identifier value (T_ID) received in the AUTH_Negotiate message, the identifiers of the hash function (HashID) and the DH group (DHgID) selected for use among the ones proposed by the host in the AUTH_Negotiate message, a sequence number (S_1), a random challenge value (C_1), and the DH exponential (g^x mod p). If the controller selects a NULL DH group identifier, then the DH portion of the DH-HMAC-CHAP protocol shall not be used, and the protocol reduces to a HMAC-CHAP transaction.

3) If the received DH-HMAC-CHAP_Challenge message is valid, then the host shall send a DH-HMAC-CHAP_Reply message (refer to section 8.13.5.4) carrying the same transaction identifier value (T_ID), the response R_1 to the challenge value C_1, and its own DH exponential (g^y mod p). The DH Value Length shall be cleared to 0h if the controller has sent a NULL DH group identifier in the DH-HMAC-CHAP_Challenge message. If bidirectional authentication is requested, then the DH-HMAC-CHAP_Reply message shall carry also a sequence number S_2 and a random challenge value C_2 that differs from the challenge value C_1 received in the DH-HMAC-CHAP_Challenge message.

4) If the authentication verification by the controller succeeds, then the controller shall reply with a DH-HMAC-CHAP_Success1 message (refer to section 8.13.5.5) carrying the same transaction identifier value (T_ID). If bidirectional authentication was requested, then the DH-HMAC-CHAP_Success1 message shall also carry the response R_2 to the challenge value C_2. If the authentication verification fails, then the controller shall send an AUTH_Failure1 message and disconnect the NVMe over Fabrics connection upon transmitting it.

5) The authentication transaction ends here, unless bidirectional authentication has been requested. In this case, as shown by the dashed arrow in Figure 443, if the authentication verification by the host succeeds, then the host shall send a DH-HMAC-CHAP_Success2 message (refer to section 8.13.5.6) carrying the same transaction identifier value (T_ID). If the authentication verification fails, then the host shall send an AUTH_Failure2 message and disconnect the NVMe over Fabrics connection upon transmitting it.

If the controller receives a message that is not the expected next message in the DH-HMAC-CHAP protocol sequence, then the controller shall:

- reply with an AUTH_Failure1 message having reason code 'Authentication failure' and reason code explanation 'Incorrect protocol message'; and
- disconnect the NVMe over Fabrics connection upon transmitting the AUTH_Failure1 message.

If the host receives a message that is not the expected next message in the DH-HMAC-CHAP protocol sequence, then the host shall:

- reply with an AUTH_Failure2 message having reason code 'Authentication failure' and reason code explanation 'Incorrect protocol message'; and
- disconnect the NVMe over Fabrics connection upon transmitting the AUTH_Failure2 message.

The payload format of a message shall be validated before performing any other security computation.

8.13.5.2 DH-HMAC-CHAP Authentication Protocol Descriptor

The authentication protocol descriptor for DH-HMAC-CHAP (refer to section 8.13.4.1) is shown in Figure 445.

Figure 445: Authentication protocol descriptor for DH-HMAC-CHAP

Bytes	Description
0	**AuthID:** Authentication protocol identifier (01h for DH-HMAC-CHAP)
1	Reserved
2	**HashIDList Length (HALEN):** Number of hash function identifiers (1 to 30)
3	**DHgIDList Length (DHLEN):** Number of Diffie-Hellman group identifiers (1 to 30)
3+HALEN:4	**HashIDList:** Array of hash function identifiers (one byte per identifier)
33:4+HALEN	Padding bytes cleared to 0h, if present
33+DHLEN:34	**DHgIDList:** Array of Diffie-Hellman Group identifiers (one byte per identifier)
63:34+DHLEN	Padding bytes cleared to 0h, if present

The one-way hash functions used by DH-HMAC-CHAP are shown in Figure 446.

Figure 446: DH-HMAC-CHAP hash function identifiers

Identifier	Hash Function	Hash Length (bytes)	Hash Block Size[1] (bytes)	Reference
00h	Reserved			
01h	SHA-256	32	64	RFC 6234
02h	SHA-384	48	128	RFC 6234
03h	SHA-512	64	128	RFC 6234
04h-DFh	Reserved			
E0h-FEh	Vendor specific			
FFh	Reserved			

Notes:
1. The hash block size is used by the HMAC calculation

The SHA-256 hash function shall be supported.

Upon receiving an AUTH_Negotiate message, if the HashIDList proposed by the host does not satisfy the security requirements of the controller (e.g., the host proposed SHA-256, but the controller's security policy requires a SHA-384 hash), then the controller shall:

- reply to the AUTH_Negotiate message with an AUTH_Failure1 message having reason code 'Authentication failure' and reason code explanation 'Hash function not usable'; and
- disconnect the NVMe over Fabrics connection upon transmitting the AUTH_Failure1 message.

The Diffie-Hellman (DH) groups used by DH-HMAC-CHAP are shown in Figure 447.

Figure 447: DH-HMAC-CHAP Diffie-Hellman group identifiers

Identifier	DH group size	Generator (*g*)	Modulus (*p*) and Reference
00h	NULL	n/a	n/a
01h	2048-bit	2	refer to RFC 7919
02h	3072-bit	2	refer to RFC 7919
03h	4096-bit	2	refer to RFC 7919
04h	6144-bit	2	refer to RFC 7919
05h	8192-bit	2	refer to RFC 7919
06h-DFh	Reserved		
E0h-FEh	Vendor specific		
FFh	Reserved		

The 00h identifier indicates that no Diffie-Hellman exchange is performed, which reduces the DH-HMAC-CHAP protocol to the HMAC-CHAP protocol.

The 2048-bit DH group and the 3072-bit DH group shall be supported. A mechanism shall be provided to disable (i.e., prohibit) use of the 2048-bit DH group.

Upon receiving an AUTH_Negotiate message, if the DHgIDList proposed by the host does not satisfy the security requirements of the controller (e.g., the host proposed only the NULL DH group, but the controller's security policy requires a DH group whose size is 3072-bit or larger), then the controller shall:

- reply to the AUTH_Negotiate message with an AUTH_Failure1 message having reason code 'Authentication failure' and reason code explanation 'DH group not usable'; and
- disconnect the NVMe over Fabrics connection upon transmitting the AUTH_Failure1 message.

8.13.5.3 DH-HMAC-CHAP_Challenge Message

The DH-HMAC-CHAP_Challenge message is sent for the controller to the host. The format of the DH-HMAC-CHAP_Challenge message is shown in Figure 448.

Figure 448: DH-HMAC-CHAP_Challenge message format

Bytes	Description
0	**AUTH_TYPE:** 01h (i.e., DH-HMAC-CHAP)
1	**AUTH_ID:** 01h (i.e., DH-HMAC-CHAP_Challenge)
3:2	Reserved
5:4	**T_ID:** 16-bit transaction identifier
6	**Hash Length (HL):** Length in bytes of the selected hash function
7	Reserved
8	**HashID:** Identifier of selected hash function
9	**DHgID:** Identifier of selected Diffie-Hellman group
11:10	**DH Value Length (DHVLEN):** Length in bytes of DH value. If no DH value is included in the message, then this field is cleared to 0h
15:12	**Sequence Number (SEQNUM):** Sequence number S_1
15+HL:16	**Challenge Value (CVAL):** Challenge C_1

Bytes	Description
15+HL+DHVLEN:16+HL	**DH Value (DHV):** DH exponential g^x mod p. This field is not present (i.e., the CVAL field is the last field in the message) if DHVLEN is cleared to 0h

Hash Length (HL): Shall be set to the length in bytes of the selected hash function, as specified in Figure 446.

HashID: Shall be set to the hash function identifier (refer to Figure 446) selected for this authentication transaction among those proposed in the DH-HMAC-CHAP protocol descriptor in the AUTH_Negotiate message. The controller shall select a hash function in accord with its applicable policy.

DHgID: Shall be set to the DH group identifier (refer to Figure 447) selected for this authentication transaction among those proposed in the DH-HMAC-CHAP protocol descriptor in the AUTH_Negotiate message. The controller shall select a DH group identifier in accord with its applicable policy. If this field is cleared to 0h, the DH portion of the DH-HMAC-CHAP protocol shall not be performed in this authentication transaction.

DH Value Length (DHVLEN): Diffie-Hellman exponential length. This length shall be a multiple of 4. If the DH group identifier is cleared to 0h (i.e., NULL DH exchange), this field shall be cleared to 0h. Otherwise, it shall be set to the length in bytes of the DH Value.

Sequence Number (SEQNUM): 32-bit sequence number S_1. A random non-zero value shall be used as the initial value. The sequence number is incremented modulo 2^{32} after each use, except that the value 0h is skipped (i.e., incrementing the value FFFFFFFFh results in the value 00000001h).

Challenge Value (CVAL): Shall be set to a random challenge value C_1 (refer to section 8.13.5.7). Each challenge value should be unique and unpredictable, since repetition of a challenge value in conjunction with the same key may reveal information about the key or the correct response to this challenge. The algorithm for generating the challenge value is outside the scope of this specification. Randomness of the challenge value is crucial to the security of the protocol (refer to section 8.13.5.7). The CVAL length is the same as the length of the selected hash function (i.e., HL).

DH Value (DHV): Diffie-Hellman exponential. If the DH Value Length is cleared to 0h, this field is not present. The DH value shall be set to the value of g^x mod p, where x is a random number selected by the controller that shall be at least 256 bits long (refer to section 8.13.5.7) and p and g shall have the values indicated in Figure 447, based on the selected DH group identifier.

Upon receiving a DH-HMAC-CHAP_Challenge message, if:

- the Hash Length (HL) does not match the value specified in Figure 446 for the selected hash function;
- the Sequence Number (SEQNUM) is cleared to 0h;
- DHgID is non-zero and the DH Value Length (DHVLEN) is cleared to 0h; or
- DHgID is non-zero and the DH Value (DHV) is 0, 1, or p-1;

then the host shall:

- reply with an AUTH_Failure2 having reason code ‘Authentication failure’ and reason code explanation ‘Incorrect payload’; and
- disconnect the NVMe over Fabrics connection.

8.13.5.4 DH-HMAC-CHAP_Reply Message

The DH-HMAC-CHAP_Reply message is sent from the host to the controller. The host may request authentication of the controller to enable bidirectional authentication, by including a DH-HMAC-CHAP challenge value C_2 in this message. The challenge value C_2 shall be different from the challenge value C_1 received in the DH-HMAC-CHAP_Challenge message.

The format of the DH-HMAC-CHAP_Reply message is shown in Figure 449.

Figure 449: DH-HMAC-CHAP_Reply message format

Bytes	Description
0	**AUTH_TYPE:** 01h (i.e., DH-HMAC-CHAP)
1	**AUTH_ID:** 02h (i.e., DH-HMAC-CHAP_Reply)
3:2	Reserved
5:4	**T_ID:** 16-bit transaction identifier
6	**Hash Length (HL):** Length in bytes of the selected hash function
7	Reserved
8	**Challenge Valid (CVALID):** Value: 00h – The Challenge Value is not valid Value: 01h – The Challenge Value is valid Value: All other values – Reserved
9	Reserved
11:10	**DH Value Length (DHVLEN):** Length in bytes of DH value. If no DH value is included in the message, then this field is cleared to 0h
15:12	**Sequence Number (SEQNUM):** Sequence number S_2
15+HL:16	**Response Value (RVAL):** Response R_1
15+2*HL:16+HL	**Challenge Value (CVAL):** Challenge C_2, if valid (i.e., if the CVALID field is set to 01h), cleared to 0h otherwise
15+2*HL+DHVLEN:16+2*HL	**DH Value (DHV):** DH exponential g^y mod p. This field is not present (i.e., the CVAL field is the last field in the message) if DHVLEN is cleared to 0h

Hash Length (HL): Shall be set to the length in bytes of the selected hash function, as specified in Figure 446.

Challenge Valid: If the host does not require bidirectional authentication or no establishment of a secure channel after unidirectional authentication is sought (refer to section 8.13), this field shall be cleared to 0h. Otherwise, this field shall be set to 01h.

DH Value Length (DHVLEN): Diffie-Hellman exponential length. This length shall be a multiple of 4. If the DH group identifier is cleared to 0h (i.e., NULL DH exchange), this field shall be cleared to 0h. Otherwise, it shall be set to the length in bytes of the DH Value.

Sequence Number (SEQNUM): 32-bit sequence number S_2. A random non-zero value shall be used as the initial value. The sequence number is incremented modulo 2^{32} after each use, except that the value 0h is skipped (i.e., incrementing the value FFFFFFFFh results in the value 00000001h). The value 0h is used to indicate that bidirectional authentication is not performed, but a challenge value C_2 is carried in order to generate a pre-shared key (PSK) for subsequent establishment of a secure channel (refer to section 8.13)

Response Value (RVAL): DH-HMAC-CHAP response value R_1. The value of R_1 is computed using the hash function H( ) selected by the HashID parameter in the DH-HMAC-CHAP_Challenge message, and the augmented challenge C_{a1}. If the NULL DH group has been selected, the augmented challenge C_{a1} is equal to the challenge C_1 received from the controller (i.e., $C_{a1} = C_1$). If a non-NULL DH group has been selected, the augmented challenge is computed applying the HMAC function using the hash function H( ) selected by the HashID parameter in the DH-HMAC-CHAP_Challenge message with the hash of the ephemeral DH key resulting from the combination of the random value y selected by the host with the DH exponential (i.e., g^x mod p) received from the controller as HMAC key (refer to RFC 2104) to the challenge C_1 (i.e., C_{a1} = HMAC(H((g^x mod p)^y mod p), C_1) = HMAC(H(g^{xy} mod p), C_1)). The value of R_1 shall be computed applying the HMAC function using the hash function H( ) selected by the HashID parameter in the DH-HMAC-CHAP_Challenge message with key K_h as HMAC key to the concatenation of the augmented challenge C_{a1}, the sequence number S_1, the transaction identifier T_ID, the secure channel concatenation indication SC_C sent in the AUTH_Negotiate message, the eight ASCII characters ”HostHost” to indicate the host is computing the reply, the host NQN not including the null terminator, a 00h

byte, and the NVM subsystem NQN not including the null terminator (i.e., R_1 = HMAC(K_h, C_{a1} || S_1 || T_ID || SC_C || "HostHost" || NQN_h || 00h || NQN_c)). Using C language notation:

C_{a1} = (DHgID == 00h) ? C_1 : HMAC(H((g^x mod p)^y mod p)), C_1)
R_1 = HMAC(K_h, C_{a1} || S_1 || T_ID || SC_C || "HostHost" || NQN_h || 00h || NQN_c)

Challenge Value (CVAL): Shall be set to a random challenge value C_2 (refer to section 8.13.5.7). Each challenge value should be unique and unpredictable, since repetition of a challenge value in conjunction with the same key may reveal information about the key or the correct response to this challenge. The algorithm for generating the challenge value is outside the scope of this specification. Randomness of the challenge value is crucial to the security of the protocol (refer to section 8.13.5.7). The CVAL length is the same as the length of the selected hash function (i.e., HL).

DH Value (DHV): Diffie-Hellman exponential. If the DH Value Length is cleared to 0h, this field is not present. The DH Value shall be set to the value of g^y mod p, where y is a random number selected by the host that shall be at least 256 bits long (refer to section 8.13.5.7) and p and g shall have the values indicated in Figure 447, based on the selected DH group identifier.

Upon receiving a DH-HMAC-CHAP_Reply message, if:

- the Hash Length (HL) does not match the value specified in Figure 446 for the selected hash function;
- DHgID is non-zero and the DH Value Length (DHVLEN) is cleared to 0h; or
- DHgID is non-zero and the DH Value (DHV) is 0, 1, or p-1;

then the controller shall:

- reply with an AUTH_Failure1 message having reason code 'Authentication failure' and reason code explanation 'Incorrect payload'; and
- disconnect the NVMe over Fabrics connection.

In addition, the controller shall:

- check the challenge value C_2, if the Challenge Valid field is set to 01h, to verify it is different from the challenge value C_1 the controller previously sent. If C_2 is equal to C_1, the controller shall:
  - reply with an AUTH_Failure1 message having reason code 'Authentication failure' and reason code explanation 'Authentication failed'; and
  - disconnect the NVMe over Fabrics connection; and
- verify the response value R_1 using the negotiated hash function. If verification of the response value R_1 does not succeed, the controller shall:
  - reply with an AUTH_Failure1 message having reason code 'Authentication failure' and reason code explanation 'Authentication failed'; and
  - disconnect the NVMe over Fabrics connection.

If verification of the response value R_1 succeeds, the host has been authenticated and the controller shall continue with a DH-HMAC-CHAP_Success1 message.

8.13.5.5 DH-HMAC-CHAP_Success1 Message

The DH-HMAC-CHAP_Success1 message is sent from the controller to the host and indicates that the controller has successfully authenticated the host. The format of the DH-HMAC-CHAP_Success1 message is shown in Figure 450.

Figure 450: DH-HMAC-CHAP_Success1 message format

Bytes	Description
0	**AUTH_TYPE:** 01h (i.e., DH-HMAC-CHAP)
1	**AUTH_ID:** 03h (i.e., DH-HMAC-CHAP_Success1)
3:2	Reserved

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>5:4</td><td>T_ID: 16-bit transaction identifier</td></tr>
<tr><td>6</td><td>Hash Length (HL): Length in bytes of the selected hash function</td></tr>
<tr><td>7</td><td>Reserved</td></tr>
<tr><td>8</td><td>Response Valid (RVALID):
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00h</td><td>The Response Value is not valid</td></tr>
<tr><td>01h</td><td>The Response Value is valid</td></tr>
<tr><td>All other values</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>15:9</td><td>Reserved</td></tr>
<tr><td>15+HL:16</td><td>Response Value (RVAL): Response R_2, if valid (i.e., if the RVALID field is set to 01h), cleared to 0h otherwise</td></tr>
</table>

Hash Length (HL): Shall be set to the length in bytes of the selected hash function, as specified in Figure 446.

Response Valid: If the host did not request authentication of the controller (i.e., bidirectional authentication) this field shall be cleared to 0h to indicate that no response is conveyed (i.e., the Response Value field is not valid). If the host did request authentication of the controller, this field shall be set to 01h.

Response Value (RVAL): DH-HMAC-CHAP response value R_2. The value of R_2 is computed using the hash function H( ) selected by the HashID parameter of the DH-HMAC-CHAP_Challenge message, and the augmented challenge C_{a2}. If the NULL DH group has been selected, the augmented challenge C_{a2} is equal to the challenge C_2 received from the host (i.e., $C_{a2} = C_2$). If a non-NULL DH group has been selected, the augmented challenge is computed applying the HMAC function using the hash function H( ) selected by the HashID parameter in the DH-HMAC-CHAP_Challenge message with the hash of the ephemeral DH key resulting from the combination of the random value x selected by the controller with the DH exponential (i.e., g^y mod p) received from the host as HMAC key (refer to RFC 2104) to the challenge C_2 (i.e., C_{a2} = HMAC(H((g^y mod p)^x mod p), C_2) = HMAC(H(g^{xy} mod p)), C_2). The value of R_2 shall be computed applying the HMAC function using the hash function H( ) selected by the HashID parameter in the DH-HMAC-CHAP_Challenge message with key K_c as HMAC key to the concatenation of the augmented challenge C_{a2}, the sequence number S_2, the transaction identifier T_ID, the secure channel concatenation indication SC_C received in the AUTH_Negotiate message, the ten ASCII characters ”Controller” to indicate the controller is computing the reply, the NVM subsystem NQN not including the null terminator, a 00h byte, and the host NQN not including the null terminator (i.e., R_2 = HMAC(K_c, C_{a2} || S_2 || T_ID || SC_C || ”Controller” || NQN_c || 00h || NQN_h)). Using C language notation:

C_{a2} = (DHgID == 00h) ? C_2 : HMAC(H((g^y mod p)^x mod p)), C_2)
R_2 = HMAC(K_c, C_{a2} || S_2 || T_ID || SC_C || ”Controller” || NQN_c || 00h || NQN_h)

Upon receiving a DH-HMAC-CHAP_Success1 message:

- if the Hash Length (HL) does not match the value specified in Figure 446 for the selected hash function, the host shall:
  - reply with an AUTH_Failure2 message having reason code ‘Authentication failure’ and reason code explanation ‘Incorrect payload’; and
  - disconnect the NVMe over Fabrics connection; and
- if the Response Valid field is set to 01h, the host shall verify the response value R_2 using the negotiated hash function and DH group. If verification of the response value R_2 does not succeed, the host shall:
  - reply with an AUTH_Failure2 message having reason code ‘Authentication failure’ and reason code explanation ‘Authentication failed’; and
  - disconnect the NVMe over Fabrics connection.

If verification of the response value R_2 succeeds, the controller has been authenticated and the host shall continue with a DH-HMAC-CHAP_Success2 message.

8.13.5.6 DH-HMAC-CHAP_Success2 Message

The DH-HMAC-CHAP_Success2 message is sent from the host to the controller and indicates that the host has successfully authenticated the controller. The format of the DH-HMAC-CHAP_Success2 message is shown in Figure 451.

Figure 451: DH-HMAC-CHAP_Success2 message format

Bytes	Description
0	**AUTH_TYPE:** 01h (i.e., DH-HMAC-CHAP)
1	**AUTH_ID:** 04h (i.e., DH-HMAC-CHAP_Success2)
3:2	Reserved
5:4	**T_ID:** 16-bit transaction identifier
15:6	Reserved

8.13.5.7 DH-HMAC-CHAP Security Requirements

In order to authenticate with the DH-HMAC-CHAP protocol, each host or controller uses a DH-HMAC-CHAP key that is associated with the entity's NQN. A DH-HMAC-CHAP key is unidirectional (i.e., used only for one direction of an authentication transaction). A DH-HMAC-CHAP key should not be associated with more than one NQN as this opens security vulnerabilities. All DH-HMAC-CHAP implementations should check for use of the same key with more than one NQN and should generate an administrative warning if this situation occurs (e.g., as a result of configuring a DH-HMAC-CHAP key to verify authentication of another entity).

The DH-HMAC-CHAP key is derived from an administratively configured secret (refer to section 8.13.5.8). Each host and NVM subsystem shall support:

- transforming the provided secret into a key applying the HMAC function using the hash function specified in the secret representation (refer to section 8.13.5.8) with the secret as HMAC key to the concatenation of its own NQN not including the null terminator and the seventeen ASCII characters "NVMe-over-Fabrics" (i.e., key = HMAC(secret, NQN || "NVMe-over-Fabrics")). This transformation ensures the resulting key is uniquely associated with the entity identified by the NQN; and
- using the provided secret as a key. This is intended for use with key management solutions able to ensure that key is uniquely associated with the entity identified by the NQN.

NVM subsystems should support the ability to use a different NVM subsystem key with each host. Hosts should support the ability to use a different host key with each NVM subsystem. NVM subsystems should support the ability to use a different NVM subsystem secret with each host. Hosts should support the ability to use a different host secret with each NVM subsystem.

If an implementation of NVMe over Fabrics is capable of functioning as both a host and an NVM subsystem, then that implementation shall use either:

- one NQN for the host functionality and a different NQN for the NVM subsystem functionality; or
- one NQN for both host functionality and NVM subsystem functionality.

DH-HMAC-CHAP implementations may reuse a DH exponential (e.g., g^x mod p or g^y mod p). The primary risk in allowing reuse of a DH exponential is replay of a prior authentication sequence based on the attacker reusing the other exponential. For DH-HMAC-CHAP, replay is prevented with extremely high probability by the requirement that all challenges be randomly generated. See section 2.12 of RFC 7296 for guidance on DH exponential reuse.

The security of the DH-HMAC-CHAP protocol requires secrets, challenges, and DH exponents (i.e., x and y) to be generated from actual randomness. For a discussion of randomness and sources of randomness, refer to RFC 4086.

Implementations shall use a cryptographic random number generator that should be seeded with at least 256 bits of entropy to generate random numbers for this protocol. The secret provisioning mechanism for

each host and controller is outside of scope of this specification. For instance, secrets could be provisioned via an encrypted HTTPS-based connection.

8.13.5.8 Secret Representation

In order to facilitate provisioning, management, and interchange (e.g., copy & paste in an administrative configuration tool) of secrets, all NVMe over Fabrics entities shall support the following ASCII representation of secrets:

```
DHHC-1:xx:<Base64 encoded string>:
```

Where:

- ”DHHC-1” indicates this is a version 1 representation of a secret for the DH-HMAC-CHAP protocol;
- ‘:’ is used both as a separator and a terminator;
- xx indicates the hash function to be used to transform the secret in key (refer to section 8.13.5.7), encoded as the ASCII representation of the hexadecimal value specified in Figure 446 (e.g., the two ASCII characters “01” indicate SHA-256). The two ASCII characters “00” indicate no transform (i.e., use the secret as a key); and
- The Base64 (refer to RFC 4648) string encodes the secret (32, 48, or 64 bytes binary) followed by the CRC-32 (refer to RFC 1952) of the secret (4 bytes binary).

As an example, the 32-byte secret:
`89AEB31A 874EAF84 841B4673 6B0DFDF2 BA58D30A A2A545A3 E235A352 1E07594Ch`
is represented as: “`DHHC-1:00:ia6zGodOr4SEG0Zzaw398rpY0wqipUWj4jWjUh4HWUz6aQ2n:`”
when intended to be used as a key without transform.

When provided with a secret in this format, NVMe over Fabrics entities shall verify the validity of the provided secret by computing the CRC-32 value of the secret and checking the computed value with the provided value. If they do not match, then the secret shall not be used.

8.13.5.9 Generated PSK for TLS

When used with a non-NULL DH exchange, the DH-HMAC-CHAP protocol is able to generate a session key K_S used to generate a pre-shared key (PSK) to establish a secure channel session with the TLS protocol between host and controller. A TLS session is concatenated to an authentication transaction when the SC_C indication is set to 01h in the AUTH_Negotiate message. A TLS session should not be concatenated to an authentication transaction if the involved host and controller are administratively configured with a PSK for use with each other. In this case, host and controller should just establish a TLS session based on the configured PSK.

The session key K_S shall be computed from the ephemeral DH key (i.e., g^{xy} mod p) generated during the DH-HMAC-CHAP transaction by applying the hash function H( ) selected by the HashID parameter in the DH-HMAC-CHAP_Challenge message (i.e., $K_S = H(g^{xy} \bmod p)$). The size of the session key K_S is determined by the selected hash function, as shown in Figure 446. Specifically:

- The host computes K_S as the hash of the ephemeral DH key resulting from the combination of the random value y selected by the host with the DH exponential (i.e., g^x mod p) received from the controller (i.e., $K_S = H((g^x \bmod p)^y \bmod p) = H(g^{xy} \bmod p)$).
- The controller computes K_S as the hash of the ephemeral DH key resulting from the combination of the random value x selected by the controller with the DH exponential (i.e., g^y mod p) received from the host (i.e., $K_S = H((g^y \bmod p)^x \bmod p) = H(g^{xy} \bmod p)$).

The generated PSK for TLS shall be computed applying the HMAC function using the hash function H( ) selected by the HashID parameter in the DH-HMAC-CHAP_Challenge message with the session key K_S as key to the concatenation of the two challenges C_1 and C_2 (i.e., generated PSK = HMAC(K_S, C_1 || C_2)). The generated PSK used to set up a TLS secure channel on the Admin Queue may be reused to set up additional TLS secure channels on the I/O queues (refer to the PSK Reuse section of the NVMe TCP Transport Specification). The lifetime of this generated PSK should be no more than ten minutes; this requires authentication for I/O queues created after this time.

The host may request secure channel concatenation with the TLS protocol by setting the SC_C indication in the AUTH_Negotiate message to 01h while performing only unidirectional authentication. In this case the host shall still send a challenge value C_2 to the controller and clear the sequence number S_2 to 0h to indicate that controller authentication is not requested.

8.13.5.10 Mapping of DH-HMAC-CHAP Messages to Authentication Commands

The DH-HMAC-CHAP_Reply message and the DH-HMAC-CHAP_Success2 message are sent from the host to the controller, therefore they are mapped to the Authentication Send command. The DH-HMAC-CHAP_Challenge message and the DH-HMAC-CHAP_Success1 message are sent from the controller to the host, therefore they are mapped to the Authentication Receive command.

8.14 Persistent Memory Region

The Persistent Memory Region (PMR) is an optional region of general purpose PCI Express read/write persistent memory that may be used for a variety of purposes. The controller indicates support for the PMR by setting CAP.PMRS (refer to section 3.1.3.1) to '1' and indicates whether the controller supports command data and metadata transfers to or from the PMR by setting support flags in the PMRCAP property. When command data and metadata transfers to or from PMR are supported, all data and metadata associated with a particular command shall be either entirely located in the Persistent Memory Region or outside the Persistent Memory Region.

The PMR's PCI Express address range is used for external memory read and write requests to the PMR. The PCI Express address range and size of the PMR is defined by the PCI Base Address Register (BAR) indicated by PMRCAP.BIR. The PMR consumes the entire address region exposed by the BAR and supports all the required features of the PCI Express programming model (i.e., it in no way restricts what is otherwise permitted by PCI Express).

The controller uses the PMR's controller address range to reference PMR with addresses supplied by the host. The PCI Express address range and the controller address range of the PMR may differ, but both ranges have the same size, and equivalent offsets within each range have a one-to-one correspondence. The host configures the controller address range via the PMRMSCU and PMRMSCL properties.

The host enables the PMR's controller memory space via the PMRMSCL.CMSE bit. When controller memory space is enabled, if host supplies an address referencing the PMR's controller address range, then the controller directs memory read or write requests for this address to the PMR.

When the PMR's controller memory space is disabled, the controller does not consider any host-supplied address to reference the PMR's controller address range, and memory read and write requests are directed elsewhere (e.g., to memory other than the PMR).

The contents of data written to the PMR while the PMR is ready persists across power cycles, Controller Level Resets, and disabling of the PMR. The mechanism used to make a write to the PMR persistent is implementation specific. For example, in one implementation this may mean that a write to non-volatile memory has completed while in another implementation this may mean that the write has been stored in a non-volatile write buffer and is written to non-volatile memory at some later point.

A PMR implementation has a maximum sustained write throughput. The PMR implementation may also have an optional write elasticity buffer used to buffer writes from PMR PCIe write requests. When the PMR sustained write throughput is less than the PCI Express link throughput, then such a write elasticity buffer allows PCIe write request burst throughput to exceed the PMR sustained write throughput without back pressuring into the PCI Express fabric.

The time required to transfer data from the write elasticity buffer to nonvolatile media is the amount of data written to the elasticity buffer divided by the Persistent Memory Region Sustained Write Throughput (refer to section 3.1.3.26). The time to transfer the entire contents of the write elasticity buffer is the Persistent Memory Region Elasticity Buffer Size (refer to section 3.1.3.25) divided by the Persistent Memory Region Sustained Write Throughput.

The host enables the PMR by setting PMRCTL.EN to '1'. Once enabled, the controller indicates that the PMR is ready by clearing PMRSTS.NRDY to '0'. It is not necessary to enable the controller to enable the

PMR. Restoring and saving the contents of the PMR may take time to complete. When the host modifies the value of PMRCTL.EN, the host should wait for at least the time interval specified in PMRCAP.PMRTO for PMRSTS.NRDY to reflect the change.

When the PMR is not ready, PMR reads complete successfully and return an undefined value while PMR writes complete normally, but do not update memory (i.e., the contents of the PMR address written remains unchanged). The undefined value returned by a PMR read following a sanitize operation is such that recovery of any previous user data from any cache or the non-volatile media is not possible.

When the PMR becomes read-only or unreliable, then a critical warning is reported in the SMART/Health Information Log which may be used to trigger an NVMe interface asynchronous event. Since reporting of an asynchronous event may occur an unspecified amount of time after the PMR health status has changed, the host should assume that all operations to the PMR have been affected since the last time normal operation was reported in PMRSTS.HSTS.

PMRCAP.PMRWBM enumerates supported PMR write barrier mechanisms. At least one mechanism shall be supported. An implementation may optionally support a mechanism where a PCI Express read of any size to the PMR, including a “zero-length read,” ensures that all previous memory writes (i.e., Posted PCI Express requests) to the PMR have completed and are persistent. An implementation may optionally support a write barrier mechanism that utilizes a read of the PMRSTS property. When supported, a read of the PMRSTS property allows a host to:

- ensure that previously issued memory writes to the PMR have completed; and
- determine whether the PMR updates associated with those writes have completed without error and are persistent.

A PMR memory write error may be the result of a poisoned PCI Express TLP, an NVM subsystem internal error, or a PMR health status issue.

Regardless of the supported PMR write barrier mechanisms, a host may periodically read the PMRSTS property to ensure that reads to the PMR have returned valid data. For example, if a read to the PMRSTS property indicates that the PMR is operating normally is then followed by a series of reads, and finally a second read to the PMRSTS property that indicates the PMR is unreliable, then one or more of the reads between the two PMRSTS property reads may have returned invalid data. Such polling of the PMRSTS property may be unnecessary if the host handles poisoned TLPs and/or poisoned TLP error reporting is enabled.

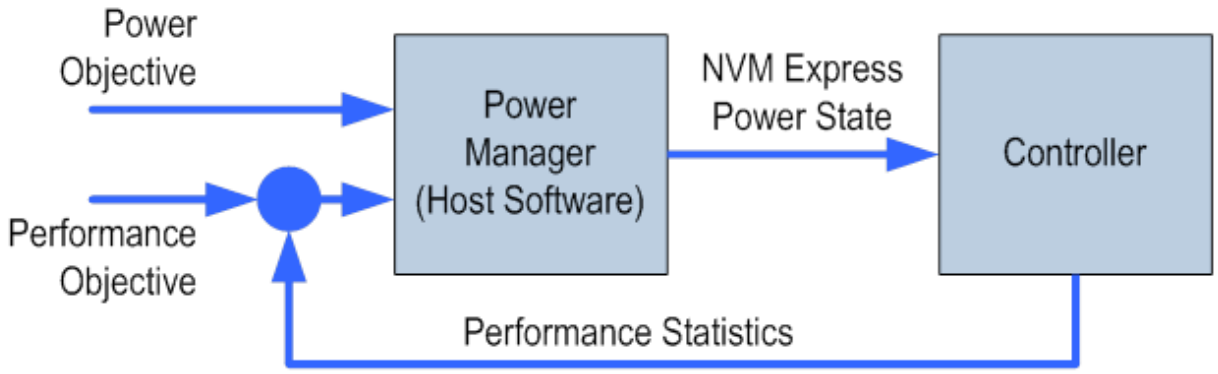
The PMR write elasticity buffer size along with the PMR sustained write throughput allows a host to determine the amount of time for a read associated with a Persistent Memory Region write barrier mechanism to complete.

Support for PRPs, SGL Lists, Completion Queues, and Submission Queues in the Persistent Memory Region is outside the scope of this specification. If the host attempts to use the Persistent Memory Region for a PRP, SGL List, Completion Queue, or Submission Queue, the controller may abort the command with a status code of Invalid Field in Command.

8.15 Power Management

The power management capability allows the host to manage NVM subsystem power statically or dynamically. Static power management consists of the host determining the maximum power that may be allocated to an NVM subsystem and setting the NVM Express power state to one that consumes this amount of power or less. Dynamic power management is illustrated in Figure 452 and consists of the host modifying the NVM Express power state to best satisfy changing power and performance objectives. This power management mechanism is meant to complement and not replace autonomous power management or thermal management performed by a controller.

Figure 452: Dynamic Power Management



The number of power states implemented by a controller is returned in the Number of Power States Supported (NPSS) field in the Identify Controller data structure. If the controller supports this feature, at least one power state shall be defined and optionally, up to a total of 32 power states may be supported. Power states shall be contiguously numbered starting with zero such that each subsequent power state consumes less than or equal to the maximum power consumed in the previous state. Thus, power state zero indicates the maximum power that the NVM subsystem is capable of consuming.

Associated with each power state is a Power State Descriptor in the Identify Controller data structure (refer to Figure 276). The descriptors for all implemented power states may be viewed as forming a table as shown in the example in Figure 453 for a controller with seven implemented power states. Note that Figure 453 is illustrative and does not include all fields in the power state descriptor. The Maximum Power (MP) field indicates the sustained maximum power that may be consumed in that state, where power measurement methods are outside the scope of this specification. The controller may employ autonomous power management techniques to reduce power consumption below this level, but under no circumstances is power allowed to exceed this level except for non-operational power states as described in section 8.15.1.

Figure 453: Example Power State Descriptor Table

Power State	Maximum Power (MP)	Entry Latency (ENLAT)	Exit Latency (EXLAT)	Relative Read Throughput (RRT)	Relative Read Latency (RRL)	Relative Write Throughput (RWT)	Relative Write Latency (RWL)
0	25 W	5 µs	5 µs	0	0	0	0
1	18 W	5 µs	7 µs	0	0	1	0
2	18 W	5 µs	8 µs	1	0	0	0
3	15 W	20 µs	15 µs	2	0	2	0
4	10 W	20 µs	30 µs	1	1	3	0
5	8 W	50 µs	50 µs	2	2	4	0
6	5 W	20 µs	5,000 µs	4	3	5	1

The Idle Power (IDLP) field indicates the typical power consumed by the NVM subsystem over 30 seconds in the power state when idle (e.g., there are no pending commands, property accesses, background processes, nor device self-test operations). The measurement starts after the NVM subsystem has been idle for 10 seconds.

The Active Power (ACTP) field indicates the largest average power of the NVM subsystem over a 10 second window on a particular workload (refer to section 8.15.3). Active Power measurement starts when the first command is submitted and ends when the last command is completed. The largest average power over a 10 second window, consumed by the NVM subsystem in that state is reported in the Active Power field. If the workload completes faster than 10 seconds, the average active power should be measured over the period of the workload. Non-operational states shall set Active Power Scale, Active Power Workload, and Active Power fields to 0h.

The host may dynamically modify the power state using the Set Features command and determine the current power state using the Get Features command. The host may directly transition between any two supported power states. The Entry Latency (ENLAT) field in the Power State Descriptor data structure indicates the maximum amount of time in microseconds to enter that power state and the Exit Latency (EXLAT) field indicates the maximum amount of time in microseconds to exit that state.

The maximum amount of time to transition between any two power states is equal to the sum of the old state's exit latency and the new state's entry latency. The host is not required to wait for a previously submitted power state transition to complete before initiating a new transition. The maximum amount of time for a sequence of power state transitions to complete is equal to the sum of transition times for each individual power state transition in the sequence.

Associated with each power state descriptor are Relative Read Throughput (RRT), Relative Write Throughput (RWT), Relative Read Latency (RRL) and Relative Write Latency (RWL) fields that provide the host with an indication of relative performance in that power state. Relative performance values provide an ordering of performance characteristics between power states. Relative performance values may repeat, may be skipped, and may be assigned in any order (i.e., increasing power states are not required to have increasing relative performance values).

A lower relative performance value indicates better performance (e.g., higher throughput or lower latency). For example, in Figure 453 power state 1 has higher read throughput than power state 2, and power states 0 through 3 all have the same read latency. Relative performance ordering is only with respect to a single performance characteristic. Thus, although the relative read throughput value of one power state may equal the relative write throughput value of another power state, this does not imply that the actual read and write performance of these two power states are equal.

The default NVM Express power state is implementation specific and shall correspond to a state that does not consume more power than the lowest value specified in the applicable form factor specification, if any. Refer to the Power Management section in the applicable NVMe Transport binding specification for transport specific power requirements impacting NVMe power states, if any.

8.15.1 Non-Operational Power States

A power state may be a non-operational power state, as indicated by Non-Operational State (NOPS) field in Figure 276. Non-operational power states allow the following operations:

- property accesses;
- PMR accesses, if any;
- CMB accesses, if any;
- processing of Admin commands and processing background operations, if any, initiated by that command (e.g., Device Self-test command (refer to section 5.9), Sanitize command (refer to section 5.24)); and
- additional transport-specific accesses as defined in the applicable NVMe Transport binding specification.

For the operations listed in the preceding paragraph, the controller:

- may exceed the power advertised by the non-operational power state;
- shall logically remain in the current non-operational power state unless an I/O command is received or if an explicit transition is requested by a Set Features command with the Power Management Feature Identifier; and
- shall not exceed the maximum power advertised for the most recent operational power state.

Execution of controller initiated background operations may exceed the power advertised by the non-operational power state, if Non-Operational Power State Permissive Mode is supported and enabled (refer to section 5.27.1.14).

No I/O commands are processed by the controller while in a non-operational power state. The host should wait until there are no pending I/O commands prior to issuing a Set Features command to change the current power state of the device to a non-operational power state and not submit new I/O commands until

the Set Features command completes. Issuing an I/O command in parallel may result in the controller being in an unexpected power state.

When in a non-operational power state, regardless of whether autonomous power state transitions are enabled, the controller shall autonomously transition back to the most recent operational power state to process an I/O command.

8.15.2 Autonomous Power State Transitions

The controller may support autonomous power state transitions, as indicated in the Identify Controller data structure in Figure 275. Autonomous power state transitions provide a mechanism for the host to configure the controller to automatically transition between power states on certain conditions without software intervention.

The entry condition to transition to the Idle Transition Power State is that the controller has been in idle for a continuous period of time exceeding the Idle Time Prior to Transition time specified. The controller is idle when there are no commands outstanding to any I/O Submission Queue. If a controller has an operation in process (e.g., device self-test operation) that would cause controller power to exceed that advertised for the proposed non-operational power state, then the controller should not autonomously transition to that state.

The power state to transition to shall be a non-operational power state (a non-operational power state may autonomously transition to another non-operational power state). If an operational power state is specified, then the controller should abort the command with a status code of Invalid Field in Command. Refer to section 8.15.1 for more details.

8.15.3 NVM Subsystem Workloads

The workload values described in this section may specify a workload hint in the Power Management Feature (refer to section 5.27.1.2) to inform the NVM subsystem or indicate the conditions for the active power level.

Active power values in the power state descriptors are specified for a particular workload since they may vary based on the workload of the NVM subsystem. The workload field indicates the conditions to observe the energy values. If Active Power is indicated for a power state, a corresponding workload shall also be indicated.

The workload values are described in Figure 454.

Figure 454: Workload Hints

Value	Description
000b	**No Workload:** The workload is unknown or not provided.
001b	**Workload #1:** Extended Idle Period with a Burst of Random Writes. Workload #1 consists of five (5) minutes of idle followed by thirty-two (32) random write commands of size 1 MiB submitted to a single controller while all other controllers in the NVM subsystem are idle, and then thirty (30) seconds of idle.
010b	**Workload #2:** Heavy Sequential Writes. Workload #2 consists of 80,000 sequential write commands of size 128 KiB submitted to a single controller while all other controllers in the NVM subsystem are idle. The submission queue(s) should be sufficiently large allowing the host to ensure there are multiple commands pending at all times during the workload.
011b to 111b	Reserved

8.15.4 Runtime D3 Transitions

In Runtime D3 (RTD3) main power is removed from the controller. Auxiliary power may or may not be provided. RTD3 is used for additional power savings when the controller is expected to be idle for a period of time.

To enable host software to determine when to use RTD3, the controller reports the latency to enter RTD3 and the latency to resume from RTD3 in the Identify Controller data structure in Figure 275. The host may

use the sum of these two values to evaluate whether the expected idle period is long enough to benefit from a transition to RTD3.

The RTD3 Resume Latency is the expected elapsed time from the time power is applied until the controller is able to:

a) process and complete I/O commands; and
b) access the NVM associated with attached namespace(s), if any, as part of I/O command processing.

The latency reported is based on a normal shutdown with optimal controller settings preceding the RTD3 resume. The latency reported assumes that host software enables and initializes the controller and sends a 4 KiB read operation.

If CSTS.ST is cleared to '0', then the RTD3 Entry Latency is the expected elapsed time from the time CC.SHN is set to 01b by host software until CSTS.SHST is set to 10b by the controller. When CSTS.SHST is set to 10b, it is safe for host software to remove power from the controller.

In this specification, RTD3 refers to the $D3_{cold}$ power state described in the PCI Express Specification. RTD3 does not include the PCI Express $D3_{hot}$ power state because main power is not removed from the controller in the $D3_{hot}$ power state. Refer to the PCI Express Base Specification for details on the $D3_{hot}$ power state and the $D3_{cold}$ power state.

8.15.5 Host Controlled Thermal Management

A controller may support host controlled thermal management (HCTM), as indicated in the Host Controlled Thermal Management Attributes of the Identify Controller data structure in Figure 275. Host controlled thermal management provides a mechanism for the host to configure a controller to automatically transition between active power states or perform vendor specific thermal management actions in order to attempt to meet thermal management requirements specified by the host. If active power states transitions are used to attempt to meet these thermal management requirements specified by the host, then those active power states transitions are vendor specific.

The host specifies and enables the thermal management requirements by setting the Thermal Management Temperature 1 field and/or Thermal Management Temperature 2 field (refer to section 5.27.1.13) in a Set Features command to a non-zero value. The supported range of values for the Thermal Management Temperature 1 field and Thermal Management Temperature 2 field are indicated in the Identify Controller data structure in Figure 275.

The Thermal Management Temperature 1 specifies that if the Composite Temperature (refer to Figure 208) is:

a) greater than or equal to this value; and
b) less than the Thermal Management Temperature 2, if non-zero,

then the controller should start transitioning to lower power active power states or perform vendor specific thermal management actions while minimizing the impact on performance in order to attempt to reduce the Composite Temperature (e.g., transition to an active power state that performs light throttling).

The Thermal Management Temperature 2 field specifies that if the Composite Temperature is greater than or equal to this value, then the controller shall start transitioning to lower power active power states or perform vendor specific thermal management actions regardless of the impact on performance in order to attempt to reduce the Composite Temperature (e.g., transition to an active power state that performs heavy throttling).

If the controller is currently in a lower power active power state or performing vendor specific thermal management actions because of this feature (e.g., throttling performance) because the Composite Temperature is:

a) greater than or equal to the current value of the Thermal Management Temperature 1 field; and
b) less than the current value of the Thermal Management Temperature 2 field,

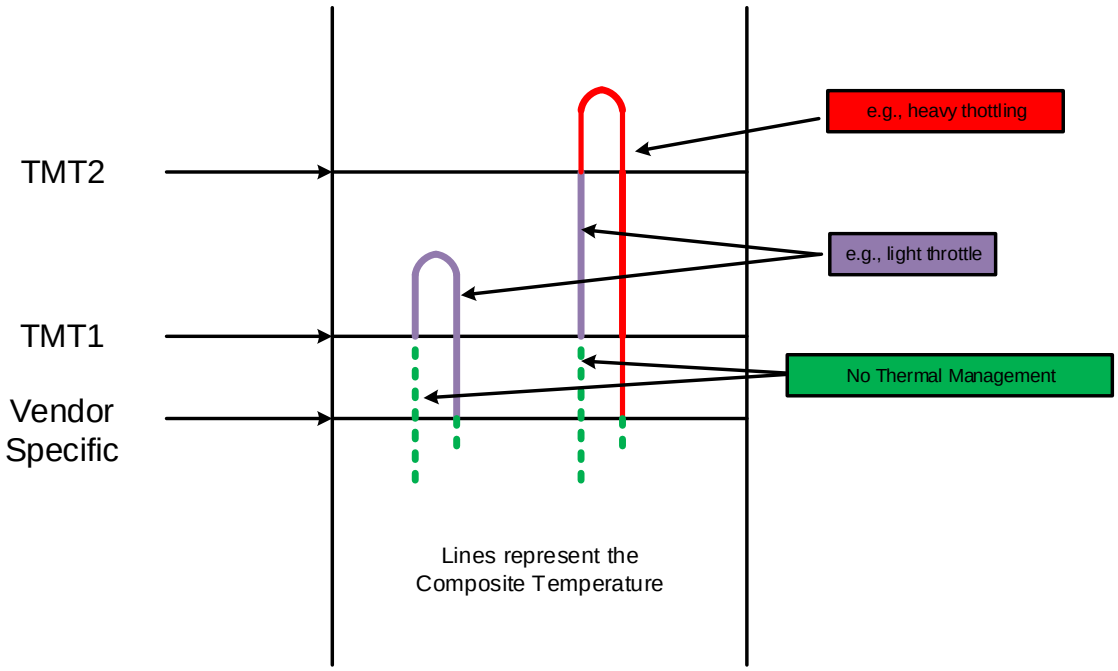
and the Composite Temperature decreases to a value below the current value of the Thermal Management Temperature 1 field, then the controller should return to the active power state that the controller was in prior to going to a lower power active power state or stop performing vendor specific thermal management actions because of this feature, the Composite Temperature and the current value of the Thermal Management Temperature 1 field.

If the controller is currently in a lower power active power state or performing vendor specific thermal management actions because the Composite Temperature is greater than or equal to the current value of the Thermal Management Temperature 2 field and the Composite Temperature decreases to a value less than the current value of the Thermal Management Temperature 1 field, then the controller should return to the active power state that the controller was in prior to going to a lower power active power state or stop performing vendor specific thermal management actions because of this feature, and the Composite Temperature.

The temperature at which the controller stops being in a lower power active power state or performing vendor specific thermal management actions because of this feature is vendor specific (i.e., hysteresis is vendor specific).

Figure 455 shows examples of how the Composite Temperature may be affected by this feature.

Figure 455: HCTM Example



Note: Since the host controlled thermal management (HCTM) feature uses the Composite Temperature, the actual interactions between a platform (e.g., tablet, or laptop) and two different device implementations may vary even with the same Thermal Management Temperature 1 and Thermal Management Temperature 2 temperature settings. The use of this feature requires validation between those devices' implementations and the platform in order to be used effectively.

8.16 Predictable Latency Mode

Predictable Latency Mode is used to achieve predictable latency for read and write operations. When configured to operate in this mode using the Predictable Latency Mode Config Feature (refer to section 5.27.1.16), the namespaces in an NVM Set (refer to section 3.2.2) provide windows of operation for deterministic operation or non-deterministic operation.

When Predictable Latency Mode is enabled:

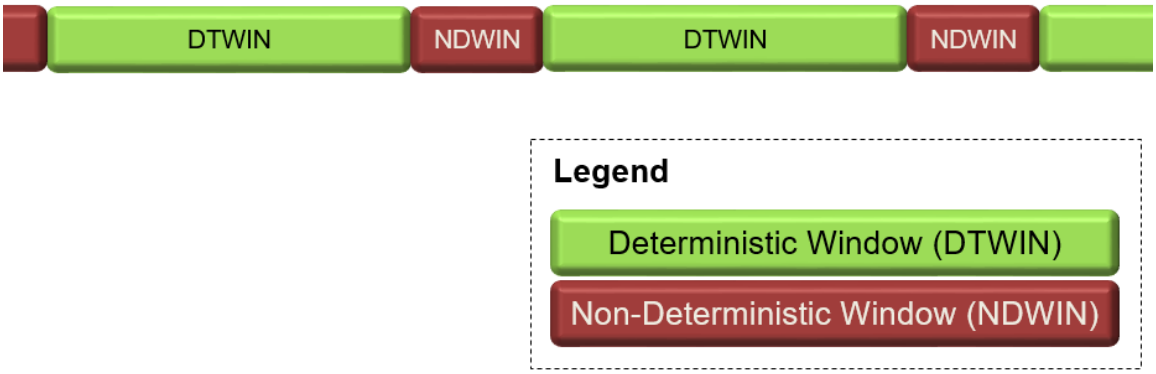
- NVM Sets and their associated namespaces have vendor specific quality of service attributes;
- I/O commands that access NVM in the same NVM Set have the same quality of service attributes; and
- I/O commands that access NVM in one NVM Set do not impact the quality of service of I/O commands that access NVM in a different NVM Set.

The quality of service attributes apply within the NVM subsystem and do not include the PCIe or fabric connection. To enhance isolation, the host should submit I/O commands for different NVM Sets to different I/O Submission Queues.

Read Recovery Levels (refer to section 8.17) shall be supported when Predictable Latency Mode is supported. The host configures the Read Recovery Level to specify the tradeoff between the quality of service versus the amount of error recovery to apply for a particular NVM Set.

The Deterministic Window (DTWIN) is the window of operation during which the NVM Set is able to provide deterministic latency for read and write operations. The Non-Deterministic Window (NDWIN) is the window of operation during which the NVM Set is not able to provide deterministic latency for read and write operations as a result of preparing for a subsequent Deterministic Window. Examples of actions that may be performed in the Non-Deterministic Window include background operations on the non-volatile media. The current window that an NVM Set is operating in is configured by the host using the Predictable Latency Mode Window Feature or by the controller as a result of an autonomous action.

Figure 456: Deterministic and Non-Deterministic Windows



To remain in the Deterministic Window, the host is required to follow operating rules (refer to section 8.16.1) ensuring that certain attributes do not exceed the typical or maximum values indicated in the Predictable Latency Per NVM Set log page. If the attributes exceed any of the typical or maximum values indicated in the Predictable Latency Per NVM Set log page or a Deterministic Excursion occurs, then the associated NVM Set may autonomously transition to the Non-Deterministic Window. A Deterministic Excursion is a rare occurrence in the NVM subsystem that requires immediate action by the controller.

The host configures Predictable Latency Events to report using the Predictable Latency Mode Config feature. The host may configure a Predictable Latency Event to be triggered when that value exceeds a specific value in order to manage window changes and avoid autonomous transitions by the controller. Refer to section 8.16.3.

If Predictable Latency Mode is supported, then all controllers in the NVM subsystem shall:

- Support one or more NVM Sets;
- Support Read Recovery Levels;
- Support the Predictable Latency Mode log page for each NVM Set;
- Support the Predictable Latency Event Aggregate log page;
- Support the Predictable Latency Mode Config Feature;
- Support the Predictable Latency Mode Window Feature;
- Support Predictable Latency Event Aggregate Log Change Notices; and
- Indicate support for Predictable Latency Mode in the Controller Attributes field in the Identify Controller data structure.

8.16.1 Host Operating Rules to Achieve Determinism

In order to achieve deterministic operation, the host is required to follow operating rules.

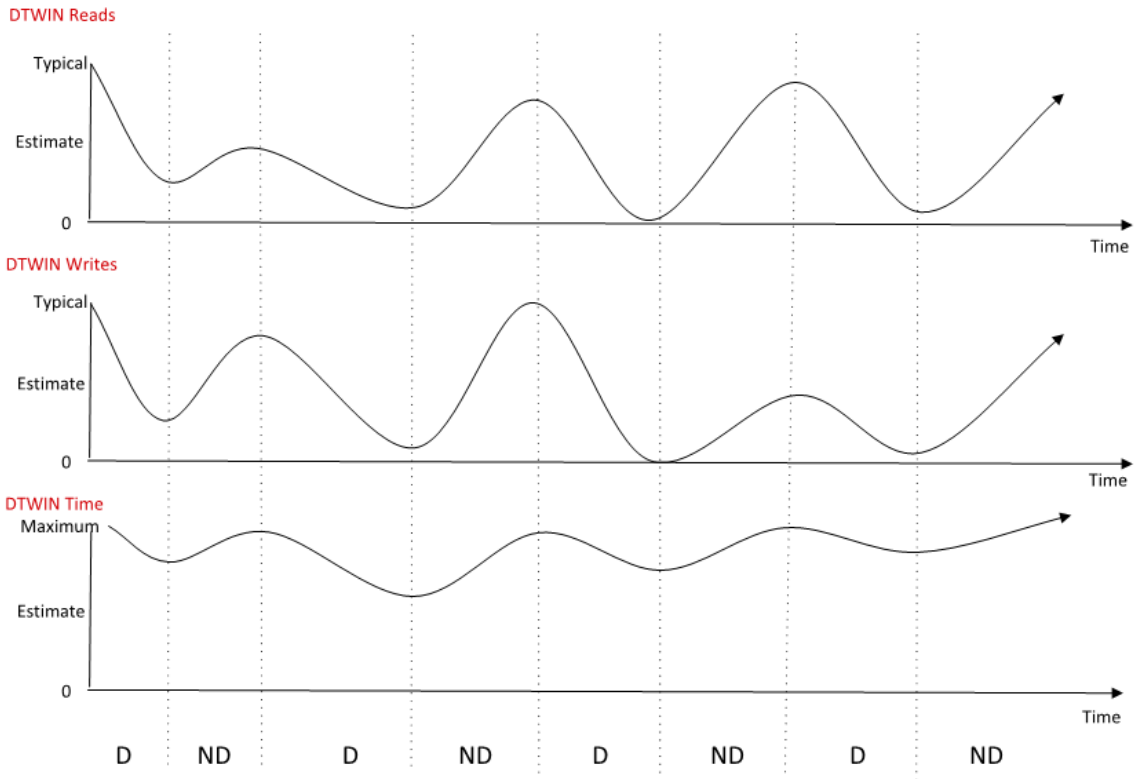
An NVM Set remains in the Deterministic Window while attributes do not exceed any of the typical or maximum values indicated in the Predictable Latency Per NVM Set log page, there is not a Deterministic Excursion, and the host does not request a transition to the Non-Deterministic Window. The attributes specified in this specification are the number of random 4 KiB reads, the number of writes in Optimal Write Size, and time in the Deterministic Window. Additional attributes are vendor specific.

For reads, writes, and time in the Deterministic Window, two values are provided in the Predictable Latency Per NVM Set log page (refer to section 5.16.1.11):

- A typical or maximum amount of that attribute that the host may consume during any given DTWIN; and
- A reliable estimate of the amount of that attribute that remains to be consumed during the current DTWIN.

Figure 457 shows how the Typical, Maximum, and Reliable Estimates for the DTWIN attributes increase or decrease when the associated NVM Set is in the Deterministic Window or Non-Deterministic Window.

Figure 457: DTWIN Attributes and Estimates

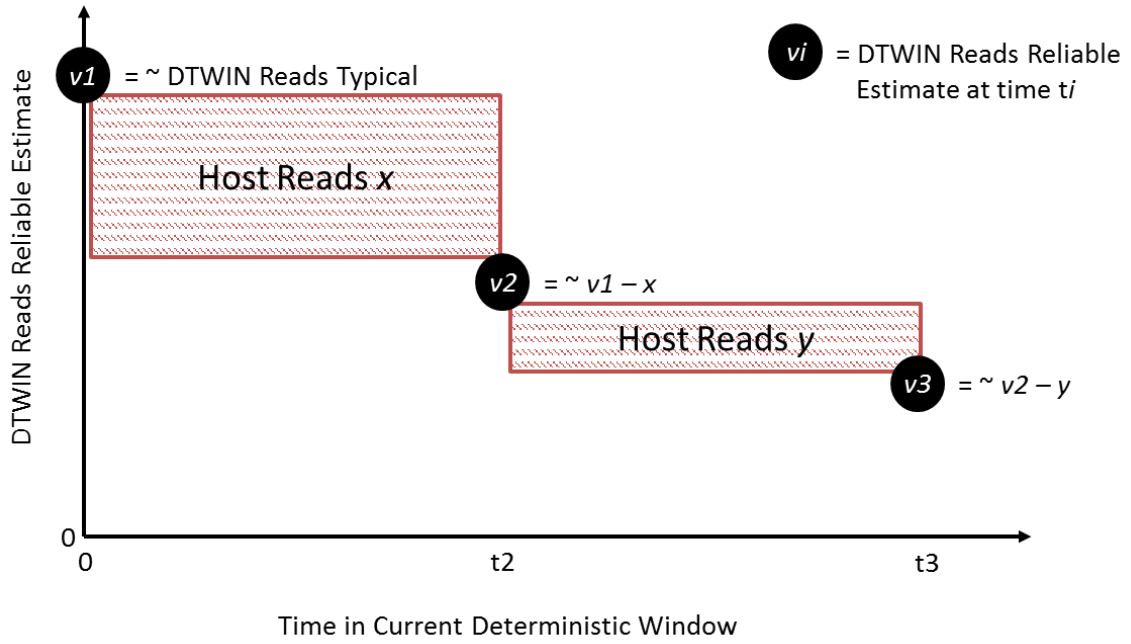


An NVM Set may transition autonomously to the NDWIN if, since entry to the current DTWIN:

a) the number of reads is greater than the value indicated in the DTWIN Reads Typical field;
b) the number of writes is greater than the value indicated in the DTWIN Writes Typical field;
c) the amount of time indicated in the DTWIN Time Maximum field has passed; or
d) a Deterministic Excursion occurs.

Figure 458 is an example that shows the relationship between the typical and reliable estimate values for DTWIN Reads. DTWIN Reads Reliable Estimate begins near the DTWIN Reads Typical value at the start of the current DTWIN at time 0. During the first time increment, the host reads *x* units, and the value of the reliable estimate at time t2 is decremented by approximately *x*. During the second time increment, the host reads a smaller amount consisting of *y* units and thus the reliable estimate at t3 is decremented by approximately *y*.

Figure 458: Typical and Reliable Estimate Example



The host configures the current window to be either DTWIN or NDWIN using a Set Features command with the Predictable Latency Mode Window Feature. The host may use the reliable estimates provided in the Predictable Latency Mode log page to ensure that the host transitions the NVM Set to the NDWIN prior to any reliable estimates exceeding one of the typical or maximum values (e.g., DTWIN Reads Estimate = 0).

The reliable estimates provided shall have the following properties when in the Deterministic Window:

- The estimates shall be monotonically decreasing towards 0h for the entirety of the DTWIN, depending on the attribute. For example, DTWIN Reads Reliable Estimate is monotonically decreasing and thus does not increase without transitioning from the DTWIN to the NDWIN; and
- The estimates shall not change abruptly unless operating conditions have changed abruptly. The estimate should be based on averaging or smoothing of data collected over some period of time.

8.16.2 Configuring Periodic Windows

When using the NVM Set in Predictable Latency Mode, the host should transition the controller to NDWIN for periodic maintenance. The maintenance is required in order for the NVM subsystem to reliably provide the amount of time indicated for Deterministic Windows.

There are three static time based parameters reported in the Predictable Latency Per NVM Set log page (refer to section 5.16.1.11) that may be used by the host to configure periodic windows. The values provided are worst-case for the life of the NVM subsystem:

- NDWIN Time Minimum Low is the minimum time that the NVM Set remains in the Non-Deterministic Window. The controller may delay completion of a Set Features command requesting a transition to the Deterministic Window until this time is completed. This time does not account for additional host activity in the Non-Deterministic Window;
- NDWIN Time Minimum High is the minimum time that the host should allow the NVM Set to remain in the Non-Deterministic Window after the NVM Set remained in the previous Deterministic Window for DTWIN Time Maximum. This time does not account for additional host activity in the Non-Deterministic Window; and
- DTWIN Time Maximum is the maximum time that the NVM Set is able to stay in a Deterministic Window.

The DTWIN Time Maximum and NDWIN Time Minimum High may provide a ratio of the amount of maintenance that needs to be performed based on the time that the NVM Set remains in the DTWIN, assuming no threshold is exceeded. Any scaling of the time in the Non-Deterministic Window based on the read, write, and time behavior in the previous Deterministic Window is implementation dependent.

The DTWIN Time Estimate may be used by the host when a Deterministic Excursion has occurred. This estimate allows the host to re-synchronize an NVM Set with other NVM Sets operating in Predictable Latency Mode, if applicable.

8.16.3 Configuring and Managing Events

The host may configure events to be triggered when thresholds do not exceed certain levels or when autonomous transitions occur using the Predictable Latency Mode Feature. The host submits a Set Feature command for the particular NVM Set and configures the specific event(s) and threshold(s) values that shall trigger a Predictable Latency Event Aggregate Log Change Notice event for that particular NVM Set to the host. Refer to Figure 348.

The host determines the NVM Sets that have outstanding events by reading the Predictable Latency Event Aggregate log page (refer to section 5.16.1.12). An entry is returned for each NVM Set that has an event outstanding. The host may use the NVM Set Identifier Maximum value reported in the Identify Controller data structure in order to determine the maximum size of this log page.

To determine the specific event(s) that have occurred for a reported NVM Sets, the host reads the Predictable Latency Per NVM Set log page (refer to section 5.16.1.11) for that NVM Set. The Event Type field indicates the event(s) that have occurred (e.g., an autonomous transition to the NDWIN). An event(s) for a particular NVM Set is cleared if the controller successfully processes a read for the Predictable Latency Per NVM Set log page for the affected NVM Set where the Get Log Page command has the Retain Asynchronous Event parameter cleared to ‘0’. If the Event Type field in the Predictable Latency Per NVM Set log page is cleared to 0h, then events for that particular NVM Set are not reported in the Predictable Latency Event Aggregate log page.

8.17 Read Recovery Level

The Read Recovery Level (RRL) is a NVM Set configurable attribute that balances the completion time for read commands and the amount of error recovery applied to those read commands. The Read Recovery Level applies to an NVM Set with which the Read Recovery Level is associated. A namespace created within an NVM Set inherits the Read Recovery Level of that NVM Set. If NVM Sets are not supported, all namespaces in the NVM subsystem use an identical Read Recovery Level.

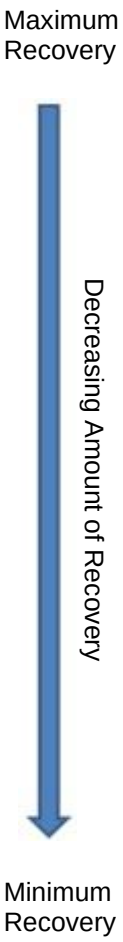
The controller indicates support for Read Recovery Levels in the Controller Attributes field in the Identify Controller data structure (refer to Figure 275). If Read Recovery Levels are supported, then the specific levels supported are indicated in the Read Recovery Levels Supported field in the Identify Controller data structure. There are 16 levels that may be supported. Level 0, if supported, provides the maximum amount of recovery. Level 4 is a mandatory level that provides a nominal amount of recovery and is the default level. Level 15 is a mandatory level that provides the minimum amount of recovery and is referred to as the ‘Fast Fail’ level. The levels are organized based on the amount of recovery supported, such that a higher numbered level provides less recovery than the preceding lower level.

Interactions between the Read Recovery Level and the Limited Retry (LR) field in I/O commands are implementation specific.

The Read Recovery Level may be configured using a Set Features command for the Read Recovery Level Config Feature. The Read Recovery Level may be determined using a Get Features command for the Read Recovery Level Config Feature.

Figure 459: Read Recovery Level Overview

Level	O/M	Description
0	O	
1	O	
2	O	
3	O	
4	M	Default
5	O	
6	O	
7	O	
8	O	
9	O	
10	O	
11	O	
12	O	
13	O	
14	O	
15	M	Fast Fail



If Read Recovery Levels are supported, then the NVM subsystem and all controllers shall:

- Support at least Level 4 and Level 15;
- Indicate support for Read Recovery Levels in the Controller Attributes field in the Identify Controller data structure;
- Support the Read Recovery Levels Supported field in the Identify Controller data structure; and
- Support the Read Recovery Level Config Feature.

8.18 Replay Protected Memory Block

The Replay Protected Memory Block (RPMB) provides a means for the system to store data to a specific memory area in an authenticated and replay protected manner. This is provided by first programming authentication key information to the controller that is used as a shared secret. The system is not authenticated in this phase, therefore the authentication key programming should be done in a secure environment (e.g., as part of the manufacturing process). The authentication key is utilized to sign the read and write accesses made to the replay protected memory area with a Message Authentication Code (MAC). Use of random number (nonce) generation and a write count property provide additional protection against replay of messages where messages could be recorded and played back later by an attacker.

The controller may support multiple RPMB targets. RPMB targets are not contained within a namespace. Controllers in the NVM subsystem may share the same RPMB targets. Security Send and Security Receive commands for RPMB do not use the namespace ID field; NSID shall be cleared to 0h. Each RPMB target

operates independently – there may be requests outstanding to multiple RPMB targets at once (where the requests may be interleaved between RPMB targets). In order to guarantee ordering the host should issue and wait for completion for one Security Send or Security Receive command at a time. Each RPMB target requires individual authentication and key programming. Each RPMB target may have its own unique Authentication Key.

The message types defined in Figure 461 are used by the host to communicate with an RPMB target. Request Message Types are sent from the host to the controller. Response Message Types are sent to the host from the controller.

Figure 460 defines the RPMB Device Configuration Block data structure – the non-volatile contents stored within the controller for RPMB target 0.

Figure 460: RPMB Device Configuration Block Data Structure

Bytes	Component Name	Description
00	Boot Partition Protection Enable	This field indicates if Boot Partition Protection is enabled. Bits 7:1 are reserved. Bit 0: A value of '1' indicates Boot Partition Protection is enabled. A value of '0' indicates Boot Partition Protection is disabled or not supported. Once enabled, the controller shall prevent disabling Boot Partition Protection
01	Boot Partition Lock	This field indicates the current status of the Boot Partition Lock. This field shall be cleared to 0h unless Boot Partition Protection is enabled. Refer to section 8.2.3. Bits 7:2 are reserved. Bit 1: A value of '1' indicates Boot Partition 1 (BPID = 1) is locked. A value of '0' indicates Boot Partition 1 (BPID = 1) is unlocked. Bit 0: A value of '1' indicates Boot Partition 0 (BPID = 0) is locked. A value of '0' indicates Boot Partition 0 (BPID = 0) is unlocked.
02	Namespace Write Protection Authentication Control	This field specifies whether the controller processes or aborts Set Features commands which enable certain namespace write protection states (refer to section 8.12 and section 5.27.1.28). If the controller does not support Namespace Write Protection, then this field shall be cleared to 0h. If the controller supports Namespace Write Protection, then bits 1:0 of this field shall be cleared to 00b after a power cycle or a Controller Level Reset. Bits 7:2 are reserved. Bit 1: If cleared to '0', indicates that the controller shall fail a Set Features command which attempts to set the namespace write protection state to Permanent Write Protect, as defined in section 8.12. If set to '1', indicates that the controller shall process a Set Features command which attempts to set the namespace write protection state to Permanent Write Protect. Bit 0: If cleared to '0', indicates that the controller shall fail a Set Features command which attempts to set the namespace write protection state to Write Protect Until Power Cycle, as defined in section 8.12. If set to '1', indicates that the controller shall process a Set Features command which sets the namespace write protection state to Write Protect Until Power Cycle.
511:03		Reserved

Each RPMB Data Frame is 256 bytes in size plus the size of the Data field, and is organized as shown in Figure 464. RPMB uses a sector size of 512 bytes. The RPMB sector size is independent and not related to the user data size used for the namespace(s).

Figure 461: RPMB Request and Response Message Types

Request Message Types		Description	Requires Data	RPMB Frame Length (bytes)
0001h	Authentication key programming request	The host is attempting to program the Authentication Key for the selected RPMB target to the controller	No	256
0002h	Reading of the Write Counter value request	The host is requesting to read the current Write Counter value from the selected RPMB target	No	256
0003h	Authenticated data write request	The host is attempting to write data to the selected RPMB target	Yes	M + 256
0004h	Authenticated data read request	The host is attempting to read data from the selected RPMB target	No	256
0005h	Result read request	The host is attempting to read the result code for any of the other Message Types	No	256
0006h	Authenticated Device Configuration Block write request	The host is attempting to write Device Configuration Block (DCB) to the selected RPMB target. This request message type is only valid for RPMB target 0.	Yes	512 + 256
0007h	Authenticated Device Configuration Block read request	The host is attempting to read Device Configuration Block (DCB) from the selected RPMB target. This request message type is only valid for RPMB target 0.	No	256
0100h	Authentication key programming response	Returned as a result of the host requesting a Result read request Message Type after programming the Authentication Key	No	256
0200h	Reading of the Write Counter value response	Returned as a result of the host requesting a Result read request Message Type after requesting the Write Counter value	No	256
0300h	Authenticated data write response	Returned as a result of the host requesting a Result read request Message Type after attempting to write data to an RPMB target	No	256
0400h	Authenticated data read response	Returned as a result of the host requesting a Result read request Message Type after attempting to read data from an RPMB target	Yes	M + 256
0600h	Authenticated Device Configuration data write response	Returned as a result of the host requesting a Result read request Message Type after attempting to write a Device Configuration Block to an RPMB target	No	256
0700h	Authenticated Device Configuration data read response	Returned as a result of the host requesting a Result read request Message Type after attempting to read DCB from an RPMB target	Yes	512 + 256

The operation result defined in Figure 462 indicates whether an RPMB request was successful or not.

Figure 462: RPMB Operation Result

Bits	Description
15:08	Reserved
07	**Write Counter Status:** Indicates if the Write Counter has expired (i.e., reached its maximum value). A value of ‘1’ indicates that the Write Counter has expired. A value of ‘0’ indicates a valid Write Counter.

Figure 462: RPMB Operation Result

Bits	Description
06:00	**Operation Status:** Indicates the operation status. Valid operation status values are listed below. **Value** / **Description** 00h – Operation successful 01h – General failure 02h – Authentication failure (MAC comparison not matching, MAC calculation failure) 03h – Counter failure (counters not matching in comparison, counter incrementing failure) 04h – Address failure (address out of range, wrong address alignment) 05h – Write failure (data/counter/result write failure) 06h – Read failure (data/counter/result read failure) 07h – Authentication Key not yet programmed. This value is the only valid Result value until the Authentication Key has been programmed. Once the key is programmed, this Result value shall no longer be used. 08h – Invalid RPMB Device Configuration Block – this may be used when the target is not 0h. 09 to 3Fh – Reserved

Figure 463 defines the non-volatile contents stored within the controller for each RPMB target.

Figure 463: RPMB Contents

Content	Type	Size	Description
Authentication Key	Write once, not erasable or readable	Size is dependent on authentication method reported in Identify Controller data structure (e.g., SHA-256 is 32 bytes (refer to RFC 6234))	Authentication key which is used to authenticate accesses when MAC is calculated.
Write Counter	Read only	4 bytes	Counter value for the total amount of successful authenticated data write requests made by the host. The initial value of this property after manufacture is 00000000h. The value is incremented by one automatically by the controller with each successful programming access. The value is not resettable. After the counter has reached the maximum value of FFFFFFFFh, the controller shall no longer increment to prevent overflow.
RPMB Data Area	Readable and writable, not erasable	Size is reported in Identify Controller data structure (128 KiB minimum, 32 MiB maximum)	Data that is able to be read and written only via successfully authenticated read/write access.

Each RPMB Data Frame is 256 bytes in size plus the size of the Data field, and is organized as shown in Figure 464. RPMB uses a sector size of 512 bytes. The RPMB sector size is independent and not related to the user data size used for the namespace(s).

Figure 464: RPMB Data Frame

Bytes	Component Name	Description
222-*N*:00	Stuff Bytes	Padding for the frame. Values in this field are not part of the MAC calculation. The size is 223 bytes minus the size of the Authentication Key (*N*).

Figure 464: RPMB Data Frame

Bytes	Component Name	Description
222:222-(N-1)	Authentication Key or Message Authentication Code (MAC)	Size is dependent on authentication method reported in the Identify Controller data structure (e.g., SHA-256 key is 32 bytes (refer to RFC 6234)).
223	RPMB Target	Indicates which RPMB this Request/Response is targeted for. Values 0-6 are supported. If the value in this field is not equal to the NVMe Security Specific Field (NSSF) in the Security Send or Security Receive command, then the controller shall return an error of Invalid Field in Command for the Security Send or Security Receive command.
239:224	Nonce	Random number generated by the host for the requests and copied to the response by the RPMB target.
243:240	Write Counter	Total amount of successfully authenticated data write requests.
247:244	Address	Starting address of data to be programmed to or read from the RPMB.
251:248	Sector Count	Number of sectors (512 bytes) requested to be read or written.
253:252	Result	Defined in Figure 462. Note: The Result field is not needed for Requests.
255:254	Request/Response Message	Defined in Figure 461.
(M-1)+256:256	Data (Optional)	Data to be written or read by signed access where *M* = 512 * Sector Count.

Security Send and Security Receive commands are used to encapsulate and deliver data packets of any security protocol between the host and controller without interpreting, dis-assembling or re-assembling the data packets for delivery. Security Send and Security Receive commands used for RPMB access are populated with the RPMB Data Frame(s) defined in Figure 464. The controller shall not return successful completion of a Security Send or Security Receive command for RPMB access until the requested RPMB Request/Response Message Type indicated is completed. The Security Protocol used for RPMB is defined in section 5.25.3.

8.18.1 Authentication Method

A controller supports one Authentication Method as indicated in the Identify Controller data structure.

If the Authentication Method supported is HMAC SHA-256 (refer to RFC 6234), then the message authentication code (MAC) is calculated using HMAC SHA-256 as defined in RFC 6234. The key used to generate a MAC using HMAC SHA-256 is the 256-bit Authentication Key stored in the controller for the selected RPMB target. The HMAC SHA-256 calculation takes as input a key and a message. Input to the MAC calculation is the concatenation of the fields in the RPMB Data Frame (request or response) excluding stuff bytes and the MAC itself – i.e., bytes [223:255] and Data of the frame in that order.

8.18.2 RPMB Operations

The host sends a Request Message Type to the controller to request an operation by the controller or to deliver data to be written into the RPMB memory block. To deliver a Request Message Type, the host uses the Security Send command. If the data to be delivered to the controller is more than reported in Identify Controller data structure, the host sends multiple Security Send commands to transfer the entire data.

The host sends a Response Message Type to the controller to read the result of a previous operation request, to read the Write Counter, or to read data from the RPMB memory block. To deliver a Response Message Type, the host uses the Security Receive command. If the data to be read from the controller is more than reported in Identify Controller data structure, the host sends multiple Security Receive commands to transfer the entire data.

8.18.2.1 Authentication Key Programming

Authentication Key programming is initiated by a Security Send command to program the Authentication Key to the specified RPMB target, followed by a subsequent Security Send command to request the result, and lastly, the host issues a Security Receive command to retrieve the result.

Figure 465: RPMB – Authentication Key Data Flow

Command	Bytes in Command	Field Name	Value	Objective
Security Send 1	**Data populated by the host and sent to the controller**			Send Authentication Key to be Programmed to the controller
	222-*N*:00	Stuff Bytes	0…00h	
	222:222-(*N*-1)	MAC/Key	*Key to be programmed*	
	223	RPMB Target	*RPMB target to access*	
	239:224	Nonce	0…00h	
	243:240	Write Counter	00000000h	
	247:244	Address	00000000h	
	251:248	Sector Count	00000000h	
	253:252	Result	0000h	
	255:254	Request/Response	0001h *(Request)*	
Security Send 2	**Data populated by the host and sent to the controller**			Request Result of Key Programming
	222-*N*:00	Stuff Bytes	0…00h	
	222:222-(*N*-1)	MAC/Key	0…00h	
	223	RPMB Target	*RPMB target to access*	
	239:224	Nonce	0…00h	
	243:240	Write Counter	00000000h	
	247:244	Address	00000000h	
	251:248	Sector Count	00000000h	
	253:252	Result	0000h	
	255:254	Request/Response	0005h *(Request)*	
Security Receive 1	**Data populated by the controller and returned to the host**			Retrieve the Key Programming Result
	222-*N*:00	Stuff Bytes	0…00h	
	222:222-(*N*-1)	MAC/Key	0…00h	
	223	RPMB Target	*RPMB target response was sent from*	
	239:224	Nonce	0…00h	
	243:240	Write Counter	00000000h	
	247:244	Address	00000000h	
	251:248	Sector Count	00000000h	
	253:252	Result	*Result Code*	
	255:254	Request/Response	0100h *(Response)*	

8.18.2.2 Read Write Counter Value

The Read Write Counter Value sequence is initiated by a Security Send command to request the Write Counter value, followed by a Security Receive command to retrieve the Write Counter result.

Figure 466: RPMB – Read Write Counter Value Flow

Command	Bytes in Command	Field Name	Value	Objective
Security Send 1	Data populated by the host and sent to the controller			Request Write Counter Read
	222-*N*:00	Stuff Bytes	0…00h	
	222:222-(*N*-1)	MAC/Key	0…00h	
	223	RPMB Target	*RPMB target to access*	
	239:224	Nonce	*Nonce generated by the host*	
	243:240	Write Counter	00000000h	
	247:244	Address	00000000h	
	251:248	Sector Count	00000000h	
	253:252	Result	0000h	
	255:254	Request/Response	0002h *(Request)*	
Security Receive 1	Data populated by the controller and returned to the host			Retrieve Write Counter Read Result
	222-*N*:00	Stuff Bytes	0…00h	
	222:222-(*N*-1)	MAC/Key	*MAC generated by the controller*	
	223	RPMB Target	*RPMB target response was sent from*	
	239:224	Nonce	*Copy of the Nonce generated by the host*	
	243:240	Write Counter	*Current Write Counter value*	
	247:244	Address	00000000h	
	251:248	Sector Count	00000000h	
	253:252	Result	*Result Code*	
	255:254	Request/Response	0200h *(Response)*	

8.18.2.3 Authenticated Data Write

The Authenticated Data Write is initiated by a Security Send command. The RPMB Data Frame delivered from the host to the controller includes the Request Message Type = 0003h, Block Count, Address, Write Counter, Data and MAC.

When the controller receives this RPMB Data Frame, that controller first checks whether the Write Counter has expired. If the Write Counter has expired, then that controller sets the result to 0005h (write failure, write counter expired) and no data is written to the RPMB data area.

After checking the Write Counter is not expired, the Address is checked. If there is an error in the Address (e.g., out of range), then the result is set to 0004h (address failure) and no data is written to the RPMB data area.

After checking the Address is valid, the controller calculates the MAC (refer to section 8.18.1) and compares this with the MAC in the request. If the MAC in the request and the calculated MAC are different, then the controller sets the result to 0002h (authentication failure) and no data is written to the RPMB data area.

If the MAC in the request and the calculated MAC are equal, then the controller compares the Write Counter in the request with the Write Counter stored in the controller. If the counters are different, then the controller sets the result to 0003h (counter failure) and no data is written to the RPMB data area.

If the MAC and Write Counter comparisons are successful, then the write request is authenticated. The Data from the request is written to the Address indicated in the request and the Write Counter is incremented by one.

If the write fails, then the returned result is 0005h (write failure). If another error occurs during the write procedure, then the returned result is 0001h (general failure).

The controller returns a successful completion for the Security Send command when the Authenticated Data Write operation is completed regardless of whether the Authenticated Data Write was successful or not.

The success of programming the data should be checked by the host by reading the result property of the RPMB:

1) The host initiates the Authenticated Data Write verification process by issuing a Security Send command with delivery of a RPMB data frame containing the Request Message Type = 0005h;
2) The controller returns a successful completion of the Security Send command when the verification result is ready for retrieval;
3) The host should then retrieve the verification result by issuing a Security Receive command; and
4) The controller returns a successful completion of the Security Receive command and returns the RPMB data frame containing the Response Message Type = 0300h, the incremented counter value, the data address, the MAC and result of the data programming operation.

Figure 467: RPMB – Authenticated Data Write Flow

Command	Bytes in Command	Field Name	Value	Objective
Security Send 1	**Data populated by the host and sent to the controller**			Program data request
	222-N:00	Stuff Bytes	0…00h	
	222:222-(N-1)	MAC/Key	MAC generated by the host	
	223	RPMB Target	RPMB target to access	
	239:224	Nonce	0…00h	
	243:240	Write Counter	Current Write Counter value	
	247:244	Address	Address in the RPMB	
	251:248	Sector Count	Number of 512B blocks	
	253:252	Result	0000h	
	255:254	Request/Response	0003h (Request)	
	(M-1)+256:256	Data	Data to be written	
Security Send 2	**Data populated by the host and sent to the controller**			Request Result of data programming
	222-*N*:00	Stuff Bytes	0…00h	
	222:222-(*N*-1)	MAC/Key	0…00h	
	223	RPMB Target	*RPMB target to access*	
	239:224	Nonce	0…00h	
	243:240	Write Counter	00000000h	
	247:244	Address	00000000h	
	251:248	Sector Count	00000000h	
	253:252	Result	0000h	
	255:254	Request/Response	0005h *(Request)*	
Security Receive 1	**Data populated by the controller and returned to the host**			Retrieve Result from data programming
	222-*N*:00	Stuff Bytes	0…00h	
	222:222-(*N*-1)	MAC/Key	*MAC generated by the controller*	
	223	RPMB Target	*RPMB target response was sent from*	
	239:224	Nonce	0…00h	
	243:240	Write Counter	*Incremented Write Counter value*	
	247:244	Address	*Address in RPMB*	
	251:248	Sector Count	00000000h	
	253:252	Result	*Result Code*	
	255:254	Request/Response	0300h *(Response)*	

8.18.2.4 Authenticated Data Read

The Authenticated Data Read sequence is initiated by a Security Send command. The RPMB data frame delivered from the host to the controller includes the Request Message Type = 0004h, Nonce, Address, and the Sector Count.

When the controller receives this RPMB Data Frame, that controller first checks the Address. If there is an error in the Address, then the result is set to 0004h (address failure) and the data read is not valid.

When the host receives a successful completion of the Security Send command from the controller, that host should send a Security Receive command to the controller to retrieve the data. The controller returns an RPMB Data Frame with Response Message Type (0400h), the Sector Count, a copy of the Nonce received in the request, the Address, the Data, the controller calculated MAC, and the Result. Note: It is the responsibility of the host to verify the MAC returned on an Authenticated Data Read Request.

If the data transfer from the addressed location in the controller fails, the returned Result is 0006h (read failure). If the Address provided in the Security Send command is not valid, then the returned Result is 0004h (address failure). If another error occurs during the read procedure, then the returned Result is 0001h (general failure).

Figure 468: RPMB – Authenticated Data Read Flow

Command	Bytes in Command	Field Name	Value	Objective
Security Send 1	**Data populated by the host and sent to the controller**			Read Data request
	222-*N*:00	Stuff Bytes	0…00h	
	222:222-(*N*-1)	MAC/Key	0..00h	
	223	RPMB Target	*RPMB target to access*	
	239:224	Nonce	*Nonce generated by the host*	
	243:240	Write Counter	00000000h	
	247:244	Address	*Address in RPMB*	
	251:248	Sector Count	*Number of 512B blocks*	
	253:252	Result	0000h	
	255:254	Request/Response	0004h *(Request)*	
Security Receive 1	**Data populated by the controller and returned to the host**			Retrieve result and data from read request
	222-*N*:00	Stuff Bytes	0…00h	
	222:222-(*N*-1)	MAC/Key	*MAC generated by the controller*	
	223	RPMB Target	*RPMB target response was sent from*	
	239:224	Nonce	*Copy of the Nonce generated by the host*	
	243:240	Write Counter	0000h	
	247:244	Address	*Address in RPMB*	
	251:248	Sector Count	*Number of 512B blocks*	
	253:252	Result	*Result Code*	
	255:254	Request/Response	0400h *(Response)*	
	(*M*-1)+256:256	Data	*Data read from RPMB target*	

8.18.3 Authenticated Device Configuration Block Write

The Authenticated Device Configuration Block Write is initiated by a Security Send command. The RPMB Data Frame delivered from the host to the controller includes the Request Message Type = 0006h, Sector Count = 01h, MAC, Write Counter set to the current Write Counter value, and the RPMB Device Configuration Block data structure (refer to Figure 469). All other fields are cleared to 0h.

If the Write Counter has expired, then that controller sets the result to 0005h (write failure, write counter expired) and no data is written to the Device Configuration Block.

The controller calculates the MAC of Request Type, Block Count, Write Counter, Address and Data, and compares this with the MAC in the request. If the MAC in the request and the calculated MAC are different, then the controller sets the result to 0002h (authentication failure) and no data is written to the RPMB Device Configuration Block.

If the Data from the RPMB Device Configuration Block attempts to disable Boot Partition Protection, then the controller sets the result to 0008h (Invalid RPMB Device Configuration Block) and no data is written to the RPMB Device Configuration Block.

If the MAC in the request and the calculated MAC are equal, then the write request is authenticated. The Data from the request is written to the RPMB Device Configuration Block.

If any other error occurs during the write procedure, then the returned result is 0001h (general failure).

The controller returns a successful completion for the Security Send command when the Authenticated Data Write operation is completed regardless of whether the Authenticated Device Configuration Block Write was successful or not.

When the host receives a successful completion of the Security Send command from the controller, that host should send a Security Receive command to the controller to retrieve the data. The controller returns an RPMB Data Frame with Response Message Type (0600h), the incremented counter value, the MAC, and the Result. All other fields are cleared to 0h.

The Write Counter for the Device Configuration Block is independent of the Write Counter for RPMB target 0. Authenticated Device Configuration Block Writes do not affect the Write Counter for RPMB target 0 since the data is not part of the RPMB data area. The current value of the Write Counter for the Device Configuration Block may be read using an Authenticated Device Configuration Block Read (refer to section 8.18.4).

Figure 469: RPMB – Authenticated Device Configuration Block Write Flow

Command	Bytes in Command	Field Name	Value	Objective
Security Send 1	**Data populated by the host and sent to the controller**			Request Device Configuration Block Write
	222-*N*:00	Stuff Bytes	0…00h	
	222:222-(*N*-1)	MAC/Key	*MAC generated by the host*	
	223	RPMB Target	00h	
	239:224	Nonce	0…00h	
	243:240	Write Counter	*Current Write Counter value*	
	247:244	Address	00000000h	
	251:248	Sector Count	00000001h	
	253:252	Result	0000h	
	255:254	Request/Response	0006h *(Request)*	
	767:256	Data	*RPMB Device Configuration Block data structure*	
Security Send 2	Data populated by the host and sent to the controller			Request Result of data programming
	222-N:00	Stuff Bytes	0…00h	
	222:222-(N-1)	MAC/Key	0…00h	
	223	RPMB Target	*RPMB target to access*	
	239:224	Nonce	0…00h	
	243:240	Write Counter	00000000h	
	247:244	Address	00000000h	
	251:248	Sector Count	00000000h	
	253:252	Result	0000h	
	255:254	Request/Response	0005h *(Request)*	
Security Receive 1	**Data populated by the controller and returned to the host**			Retrieve Device Configuration Block Write Result
	222-*N*:00	Stuff Bytes	0…00h	
	222:222-(*N*-1)	MAC/Key	*MAC generated by the controller*	
	223	RPMB Target	00h	
	239:224	Nonce	0…00h	
	243:240	Write Counter	*Incremented Write Counter value*	
	247:244	Address	00000000h	
	251:248	Sector Count	00000000h	
	253:252	Result	*Result Code*	
	255:254	Request/Response	0600h *(Response)*	

8.18.4 Authenticated Device Configuration Block Read

The Authenticated Device Configuration Block Read sequence is initiated by a Security Send command. The RPMB data frame delivered from the host to the controller includes the Nonce, Request Message Type = 0007h and the Sector Count = 01h. All other fields are cleared to 0h.

When the host receives a successful completion of the Security Send command from the controller, that host should send a Security Receive command to the controller to retrieve the data. The controller returns an RPMB Data Frame with Response Message Type (0700h), the Sector Count = 01h, a copy of the Nonce

received in the request, the RPMB Device Configuration Block Data Structure (refer to Figure 460), the MAC, the Write Counter set to the current Write Counter value, and the Result. All other fields are cleared to 0h.

The Write Counter for the Device Configuration Block is independent of the Write Counter for RPMB target 0. The controller returns the Device Configuration Block Write Counter as shown in Figure 470.

The MAC is calculated from Response Type, Nonce, Address, Data and Result fields. If the MAC calculation fails, then the returned result is 0002h (authentication failure). If another error occurs during the read procedure, then the returned Result is 0001h (general failure).

Figure 470: RPMB – Authenticated Device Configuration Block Read Flow

Command	Bytes in Command	Field Name	Value	Objective
Security Send 1	**Data populated by the host and sent to the controller**			Request Device Configuration Block Read
	222-*N*:00	Stuff Bytes	0…00h	
	222:222-(*N*-1)	MAC/Key	0..00h	
	223	RPMB Target	00h	
	239:224	Nonce	*Nonce generated by the host*	
	243:240	Write Counter	00000000h	
	247:244	Address	00000000h	
	251:248	Sector Count	00000001h	
	253:252	Result	0000h	
	255:254	Request/Response	0007h *(Request)*	
Security Receive 1	**Data populated by the controller and returned to the host**			Retrieve Device Configuration Block Read Result
	222-*N*:00	Stuff Bytes	0…00h	
	222:222-(*N*-1)	MAC/Key	*MAC generated by the controller*	
	223	RPMB Target	00h	
	239:224	Nonce	*Copy of the Nonce generated by the host*	
	243:240	Write Counter	*Current Write Counter value*	
	247:244	Address	00000000h	
	251:248	Sector Count	00000001h	
	253:252	Result	*Result Code*	
	255:254	Request/Response	0700h *(Response)*	
	767:256	Data	*RPMB Device Configuration Block data structure*	

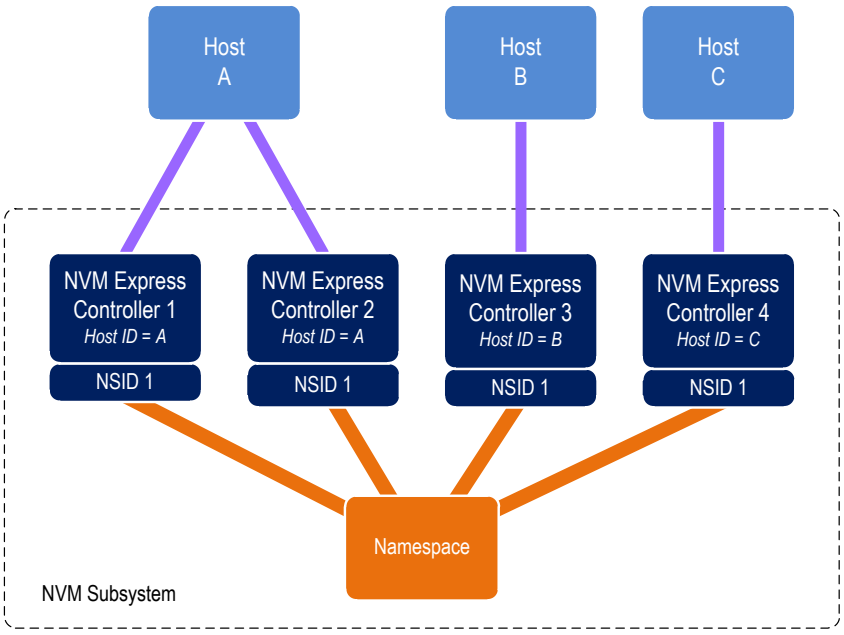
8.19 Reservations

NVM Express reservations provide capabilities that may be utilized by two or more hosts to coordinate access to a shared namespace. The protocol and manner in which these capabilities are used is outside the scope of this specification. Incorrect application of these capabilities may corrupt data and/or otherwise impair system operation.

Reservation operation after a division event (refer to section 3.2.4.1) is described in section 3.2.4.2.

A reservation on a namespace restricts hosts access to that namespace. If a host submits a command to a namespace in the presence of a reservation and lacks sufficient rights, then the command is aborted by the controller with a status code of Reservation Conflict. If a host submits a command with the NSID set to FFFFFFFFh in the presence of a reservation on any of the namespaces impacted by that command and that host lacks sufficient rights on all the impacted namespaces, then the command is aborted by the controller with a status code of Reservation Conflict. Capabilities are provided that allow recovery from a reservation on a namespace held by a failing or uncooperative host.

Figure 471: Example Multi-Host System



A reservation requires an association between a host and a namespace. As shown in Figure 471, each controller in a multi-path I/O and namespace sharing environment is associated with exactly one host. While it is possible to construct systems where two or more hosts share a single controller, such usage is outside the scope of this specification.

A host may be associated with multiple controllers. In Figure 471 host A is associated with two controllers while hosts B and C are each associated with a single controller. A host registers a Host Identifier (refer to section 5.27.1.25) with each controller with which that host is associated using a Set Features command (refer to section 5.27) prior to performing any operations associated with reservations. The Host Identifier allows the NVM subsystem to identify controllers associated with the same host and preserve reservation properties across these controllers (i.e., a host issued command has the same reservation rights no matter which controller associated with the host processes the command).

Support for reservations by a namespace or controller is optional. A namespace indicates support for reservations by reporting a non-zero value in the Reservation Capabilities (RESCAP) field in the Identify Namespace data structure. A controller indicates support for reservations through the Optional NVM Command Support (ONCS) field in the Identify Controller data structure (refer to Figure 275). If a host submits a command associated with reservations (i.e., Reservation Report, Reservation Register, Reservation Acquire, and Reservation Release) to a controller or a namespace that do not both support reservations, then the command is aborted by the controller with a status code of Invalid Command Opcode.

Controllers that make up an NVM subsystem shall all have the same support for reservations. Although strongly encouraged, namespaces that make up an NVM subsystem are not all required to have the same support for reservations. For example, some namespaces within a single controller may support reservations while others do not, or the supported reservation types may differ among namespaces. If a controller supports reservations, then the controller shall:

- Indicate support for reservations by returning a '1' in bit 5 of the Optional NVM Command Support (ONCS) field in the Identify Controller data structure;
- Support the Reservation Report command (refer to section 7.5), Reservation Register command (refer to section 7.3), Reservation Acquire command (refer to section 7.1), and Reservation Release command (refer to section 7.4);
- Support the Reservation Notification log page;
- Support the Reservation Log Page Available asynchronous events;
- Support the Reservation Notification Mask Feature;

- Support the Host Identifier Feature; and
- Support the Reservation Persistence Feature.

If a namespace supports reservations, then the namespace shall:

- Report a non-zero value in the Reservation Capabilities (RESCAP) field in the Identify Namespace data structure;
- Support Persist Through Power Loss (PTPL) state; and
- Support sufficient resources to allow a host to successfully register a reservation key on every controller in the NVM subsystem with access to the shared namespace (i.e., a Reservation Register command shall never fail due to lack of resources).

NOTE: The behavior of Ignore Existing Key has been changed to improve compatibility with SCSI based implementations. Conformance to the modified behavior is indicated in the Reservation Capabilities (RESCAP) field of the Identify Namespace data structure. For the previous definition of Ignore Existing Key behavior, refer to NVM Express Base Specification revision 1.2.1.

8.19.1 Reservation Types

The NVM Express interface supports six types of reservations:

- Write Exclusive;
- Exclusive Access;
- Write Exclusive - Registrants Only;
- Exclusive Access - Registrants Only;
- Write Exclusive - All Registrants; and
- Exclusive Access - All Registrants.

Figure 472: Command Behavior in the Presence of a Reservation

Reservation Type	Reservation Holder		Registrant		Non-Registrant		Reservation Holder Definition
	Read	Write	Read	Write	Read	Write	
Write Exclusive	Y	Y	Y	N	Y	N	One Reservation Holder
Exclusive Access	Y	Y	N	N	N	N	One Reservation Holder
Write Exclusive - Registrants Only	Y	Y	Y	Y	Y	N	One Reservation Holder
Exclusive Access - Registrants Only	Y	Y	Y	Y	N	N	One Reservation Holder
Write Exclusive - All Registrants	Y	Y	Y	Y	Y	N	All Registrants are Reservation Holders
Exclusive Access - All Registrants	Y	Y	Y	Y	N	N	All Registrants are Reservation Holders

The differences between these reservation types are: the type of access that is excluded (i.e., writes or all accesses), whether registrants have the same access rights as the reservation holder, and whether registrants are also considered to be reservation holders. These differences are summarized in Figure 472 and the specific behavior for each NVM Express command is shown in Figure 473.

Reservations and registrations persist across all Controller Level Resets and all NVM Subsystem Resets except reset due to power loss. A reservation may be optionally configured to be retained across a reset due to power loss using the Persist Through Power Loss State (PTPLS). A Persist Through Power Loss State (PTPLS) is associated with each namespace that supports reservations and may be modified as a side effect of a Reservation Register command (refer to section 7.3) or a Set Features command (refer to section 5.27).

Figure 473: Command Behavior in the Presence of a Reservation

NVMe Command	Write Exclusive Reservation		Exclusive Access Reservation		Write Exclusive Registrants Only or Write Exclusive All Registrants Reservation		Exclusive Access Registrants Only or Exclusive Access All Registrants Reservation	
	Non-Registrant	Registrant	Non-Registrant	Registrant	Non-Registrant	Registrant	Non-Registrant	Registrant
Copy Command Group								
I/O Command Set specific Copy Commands[2]	C	C	C	C	C	A	C	A
Read Command Group								
Security Receive (Admin) I/O Command Set specific Read Commands[2]	A	A	C	C	A	A	C	A
Write Command Group								
Capacity Management (Admin) Flush Format NVM (Admin) Namespace Attachment (Admin) Namespace Management (Admin) Sanitize (Admin) Security Send (Admin) I/O Command Set specific Write Commands[2]	C	C	C	C	C	A	C	A
Reservation Command Groups								
Reservation Acquire - Acquire	C	C	C	C	C	C	C	C
Reservation Acquire - Preempt Reservation Acquire - Preempt and Abort Reservation Release	C	A	C	A	C	A	C	A
All Other Commands Group								
All other commands[1]	A	A	A	A	A	A	A	A

Key:
A definition: A=Allowed, command processed normally by the controller
C definition: C=Conflict, command aborted by the controller with a status code of Reservation Conflict

Notes:
1. The behavior of a vendor specific command is vendor specific.
2. Refer to the applicable I/O Command Set specification

8.19.2 Reservation Notifications

There are three types of reservation notifications: registration preempted, reservation released, and reservation preempted. Conditions that cause a reservation notification to occur are described in the following sections. A Reservation Notification log page is created whenever an unmasked reservation notification occurs on a namespace associated with the controller (refer to section 5.16.1.24). Reservation notifications may be masked from generating a Reservation Notification log page on a per reservation notification type and per namespace ID basis through the Reservation Notification Mask feature (refer to section 5.27.1.26). A host may use the Asynchronous Event Request command (refer to section 5.2) to be notified of the presence of one or more available Reservation Notification log pages (refer to section 5.16.1.24).

8.19.3 Registering

Prior to establishing a reservation on a namespace, a host shall become a registrant of that namespace by registering a reservation key. This reservation key may be used by the host as a means of identifying the registrant (host), authenticating the registrant, and preempting a failed or uncooperative registrant. The value of the reservation key used by a host and the method used to select its value is outside the scope of this specification.

Registering a reservation key with a namespace creates an association between a host and a namespace. A host that is a registrant of a namespace may use any controller with which that host is associated (i.e., that has the same Host Identifier, refer to section 5.27.1.25) to access that namespace as a registrant. Thus, a host is only required to register on a single controller to become a registrant of the namespace on all controllers in the NVM subsystem that have access to the namespace and are associated with the host.

A host registers a reservation key by executing a Reservation Register command (refer to section 7.3) on the namespace with the Reservation Register Action (RREGA) field cleared to 000b (i.e., Register Reservation Key) and supplying a reservation key in the New Reservation Key (NRKEY) field.

A host that is a registrant of a namespace may register the same reservation key value multiple times with the namespace on the same or different controllers. For a Reservation Register command with the RREGA field cleared to 000b:

a) the IEKEY field shall be ignored; and
b) if a host that is already a registrant of a namespace attempts to register with that namespace using a different registration key value, then the command shall be aborted with a status code of Reservation Conflict.

There are no restrictions on the reservation key value used by hosts with different Host Identifiers. For example, multiple hosts may all register with the same reservation key value.

A host that is a registrant of a namespace may replace the existing reservation key value for that namespace by executing a Reservation Register command on the namespace with the:

a) RREGA field set to 010b (i.e., Replace Reservation Key);
b) current reservation key in the Current Reservation Key (CRKEY) field; and
c) new reservation key in the NRKEY field.

The current reservation key value shall be replaced by the new reservation key value in all controllers to which the namespace is attached that have the same Host Identifier as the Host Identifier of the controller processing the command. If the contents of the CRKEY field do not match the key currently associated with the host, then the command shall be aborted with a status code of Reservation Conflict. A host may replace its reservation key without regard to its registration status or current reservation key value by setting the Ignore Existing Key (IEKEY) bit to '1' in the Reservation Register command. Replacing a reservation key has no effect on any reservation that may be held on the namespace.

8.19.4 Unregistering

A host that is a registrant of a namespace may unregister with the namespace by executing a Reservation Register command (refer to section 7.3) on the namespace with the RREGA field set to 001b (i.e., Unregister Reservation Key) and supplying its current reservation key in the CRKEY field. If the contents of the CRKEY field do not match the key currently associated with the host, then the command is aborted with a status code of Reservation Conflict. If the host is not a registrant, then the command is aborted with a status code of Reservation Conflict.

Successful completion of an unregister operation causes the host to no longer be a registrant of that namespace. A host may unregister without regard to its current reservation key value by setting the IEKEY bit to '1' in the Reservation Register command.

Unregistering by a host may cause a reservation held by the host to be released. If a host is the last remaining reservation holder (i.e., the reservation type is Write Exclusive - All Registrants or Exclusive Access - All Registrants) or is the only reservation holder, then the reservation is released when the host unregisters.

If a reservation is released and the type of the released reservation was Write Exclusive - Registrants Only or Exclusive Access - Registrants Only, then a reservation released notification occurs on all controllers associated with a registered host other than the host that issued the Reservation Register command.

8.19.5 Acquiring a Reservation

In order for a host to obtain a reservation on a namespace, that host shall be a registrant of that namespace. A registrant obtains a reservation by executing a Reservation Acquire command (refer to section 7.1), clearing the Reservation Acquire Action (RACQA) field to 000b (Acquire), and supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field. The CRKEY value shall match that used by the registrant to register with the namespace. If the CRKEY value does not match, then the command is aborted with a status code of Reservation Conflict. If the host is not a registrant, then the command is aborted with a status code of Reservation Conflict.

Only one reservation is allowed at a time on a namespace. If a registrant attempts to obtain a reservation on a namespace that already has a reservation holder, then the command is aborted with a status code of Reservation Conflict. If a reservation holder attempts to obtain a reservation of a different type on a namespace for which that host already is the reservation holder, then the command is aborted with a status code of Reservation Conflict. If a reservation holder attempts to obtain a reservation of the same type on a namespace for which that host already is the reservation holder, then it is not a Reservation Conflict and the command is processed. A reservation holder may preempt a reservation to change the reservation type.

8.19.6 Releasing a Reservation

Only a reservation holder is able to release a reservation held on a namespace. A host should release a reservation using the following sequence:

a) executing a Reservation Release command (refer to section 7.4);
b) clearing the Reservation Release Action (RRELA) field to 000b (i.e., Release);
c) setting the Reservation Type (RTYPE) field to the type of reservation being released; and
d) supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field. The CRKEY value shall match that used by the host to register with the namespace.

If the key value doesn't match, then the command is aborted with a status code of Reservation Conflict. If the RTYPE field does not match the type of the current reservation, then the command completes with a status code of Invalid Field in Command.

An attempt by a registrant to release a reservation using the Reservation Release command in the absence of a reservation held on the namespace or when the host is not the reservation holder shall cause the command to complete successfully, but shall have no effect on the controller or namespace.

When a reservation is released as a result of actions described in this section and the reservation type is not Write Exclusive or Exclusive Access, a reservation released notification occurs on all controllers in the NVM subsystem that are associated with hosts that are registrants except for controllers that are associated with the host that issued the Reservation Release command.

8.19.7 Preempting a Reservation or Registration

A host that is a registrant may preempt a reservation and/or registration by executing a Reservation Acquire command (refer to section 7.1), setting the Reservation Acquire Action (RACQA) field to 001b (Preempt), and supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field. The CRKEY value shall match that used by the registrant to register with the namespace. If the CRKEY value does not match, then the command is aborted with a status code of Reservation Conflict. The preempt actions that occur are dependent on the type of reservation held on the namespace, if any, and the value of the Preempt Reservation Key (PRKEY) field in the command. If the host is not a registrant, then the command is aborted with a status code of Reservation Conflict. The remainder of this section assumes that the host is a registrant.

If the existing reservation type is not Write Exclusive - All Registrants and not Exclusive Access - All Registrants, then the actions performed by the command depend on the value of the PRKEY field as follows:

a) If the PRKEY field value matches the reservation key of the current reservation holder, then the following occur as an atomic operation:

- all registrants with a matching registration key other than the host that issued the command are unregistered;
- the reservation is released; and
- a new reservation is created of the type specified by the Reservation Type (RTYPE) field in the command for the host that issued the command as the reservation key holder;

or

b) If the PRKEY field value does not match that of the current reservation holder and is not equal to 0h, then registrants whose reservation key matches the value of the PRKEY field are unregistered. If the PRKEY field value does not match that of the current reservation holder and is equal to 0h, then the command is aborted with a status code of Invalid Field in Command.

If the existing reservation type is Write Exclusive - All Registrants or Exclusive Access - All Registrants, then the actions performed by the command depend on the value of the PRKEY field as follows:

a) If the PRKEY field value is 0h, then the following occurs as an atomic operation:

- all registrants other than the host that issued the command are unregistered;
- the reservation is released; and
- a new reservation is created of the type specified by the Reservation Type (RTYPE) field in the command for the host that issued the command as the reservation key holder;

or

b) If the PRKEY value is non-zero, then registrants whose reservation key matches the value of the PRKEY field are unregistered. If the PRKEY value is non-zero and there are no registrants whose reservation key matches the value of the PRKEY field, the controller should return an error of Reservation Conflict.

If there is no reservation held on the namespace, then execution of the command causes registrants whose reservation key match the value of the PRKEY field to be unregistered.

If the existing reservation type is not Write Exclusive - All Registrants and not Exclusive Access - All Registrants, then a reservation holder may preempt itself using the above mechanism. When a host preempts itself the following occurs as an atomic operation:

- registration of the host is maintained;
- the reservation is released; and
- a new reservation is created for the host of the type specified by the RTYPE field.

A host may abort commands as a side effect of preempting a reservation by executing a Reservation Acquire command (refer to section 7.1) and setting the RACQA field to 010b (Preempt and Abort). The behavior of such a command is exactly the same as that described above with the RACQA field set to 001b (Preempt), with two exceptions:

- After the atomic operation changes namespace reservation and registration state, all controllers associated with any host whose reservation or registration is preempted by that atomic operation are requested to abort all commands being processed that were addressed to the namespace specified in the Namespace Identifier field (i.e., the NSID field in the Reservation Acquire command) (refer to section 3.4.4 for the definition of “being processed”); and
- Completion of the Reservation Acquire command shall not occur until all commands that are requested to be aborted are completed, regardless of whether or not each command is actually aborted.

As with the Abort command (refer to section 5.1), aborting a command as a side effect of preempting a reservation is best effort; as a command that is requested to be aborted may currently be at a point in execution where that command is no longer able to be aborted or may have already completed, when a Reservation Acquire or Abort Admin command is submitted. Although prompt execution of abort requests

reduces delay in completing the Reservation Acquire command, a command which is requested to be aborted shall either be aborted or otherwise completed before the completion of the Reservation Acquire command.

When a registrant is unregistered as a result of actions described in this section, then a registration preempted notification occurs on all controllers associated with a host that was unregistered other than the host that issued the Reservation Acquire command.

When the type of reservation held on a namespace changes as a result of actions described in this section, then a reservation released notification occurs on all controllers associated with hosts that remain registrants of the namespace except the host that issued the Reservation Acquire command.

8.19.8 Clearing a Reservation

A host that is a registrant may clear a reservation (i.e., force the release of a reservation held on the namespace and unregister all registrants) by:

a) executing a Reservation Release command (refer to section 7.4);
b) setting the Reservation Release Action (RRELA) field to 001b (i.e., Clear); and
c) supplying the current reservation key associated with the host in the Current Reservation Key (CRKEY) field.

If the value in the CRKEY field does not match the value used by the host to register with the namespace, then the command shall be aborted with a status code of Reservation Conflict. If the host is not a registrant, then the command is aborted with a status code of Reservation Conflict. When a command to clear a reservation is executed the following occur as an atomic operation: the reservation held on the namespace, if any, is released, and all registrants are unregistered from the namespace.

A reservation preempted notification occurs on all controllers in the NVM subsystem that are associated with hosts that have their registrations removed as a result of actions taken in this section except those associated with the host that issued the Reservation Release command.

8.19.9 Reporting Reservation Status

A host may determine the current reservation status associated with a namespace by executing a Reservation Report command (refer to section 7.5).

8.20 Rotational Media

Rotational media has different operational, endurance and performance characteristics than non-rotational media (e.g., NAND). Rotational media utilizes electromechanical methods for accessing data.

Rotational media contains one or more spinning platters containing the media, and one or more actuators that provide physical access to the data on that media (e.g., a hard disk drive or a CD-ROM).
A controller that supports namespaces that store user data on rotational media shall:

a) set the Rotational Media bit to ‘1’ in the NSFEAT field of the I/O Command Set Independent Identify Namespace data structure (refer to the NVM Command Set Specification) for any namespace that stores data on rotational media;
b) support the Rotational Media Information log page (refer to section 5.16.1.21);
c) support the Spinup Control feature (refer to section 5.27.1.22);
d) support Endurance Groups (refer to section 3.2.3); and
e) set the EG Rotational Media bit to ‘1’ in the EGFEAT field in the Endurance Group Information log page for each Endurance Group that stores data on rotational media.

If a namespace that stores data on rotational media is attached to a controller, and the spindle used by that namespace is not spinning, then that controller shall be in a non-operational power state (i.e., NOPS is set to ‘1’, refer to Figure 276).

If:

a) a domain contains an Endurance Group that stores data on rotational meida;

b) that domain processes an NVM Subsystem Reset; and
c) the Spinup Control feature (refer to section 5.27.1.22) is:

    a. disabled, then initial spinup for all such Endurnce Groups in that domain shall be initiated; and
    b. enabled, then initial spinup for all such Endurance Groups in that domain shall be inhibited during processing of the NVM Subsystem Reset until the controller processes a Set Features (Power Management) command that specifies an operational power state.

If the PCIe transport is used for a controller, then the PCIe Slot Power Control feature may affect the power states supported.

8.21 Sanitize Operations

A sanitize operation alters all user data in the NVM subsystem such that recovery of any previous user data from any cache, the non-volatile media, or any Controller Memory Buffer is not possible. It is implementation specific whether Submission Queues and Completion Queues within a Controller Memory Buffer are altered by a sanitize operation; all other data stored in all Controller Memory Buffers is altered by a sanitize operation. If a portion of the user data was not altered and the sanitize operation completed successfully, then the NVM subsystem shall ensure permanent inaccessibility of that portion of the user data for any future use within the NVM subsystem (e.g., retrieval from NVM media, caches, or any Controller Memory Buffer) and permanent inaccessibility of that portion of the user data via any interface to the NVM subsystem, including management interfaces as defined by the NVM Express Management Interface Specification.

The scope of a sanitize operation is all locations in the NVM subsystem that are able to contain user data, including caches, Persistent Memory Regions, and unallocated or deallocated areas of the media. If the composition of the NVM subsystem (refer to section 3.2.4) changes (e.g., a new domain is added, or a division event occurs) and that change prevents the successful completion of a sanitize operation, then the sanitize operation shall fail. Sanitize operations do not affect the Replay Protected Memory Block, boot partitions, or other media and caches that do not contain user data. A sanitize operation also may alter log pages as necessary (e.g., to prevent derivation of user data from log page information). A sanitize operation is only able to be started if the NVM subsystem is not divided (refer to section 3.2.4). Once started, a sanitize operation is not able to be aborted and continues after a Controller Level Reset including across power cycles. Refer to Annex A for further information about sanitize operations.

The Sanitize command (refer to section 5.24) is used to start a sanitize operation or to recover from a previously failed sanitize operation. All sanitize operations are performed in the background (i.e., completion of the Sanitize command does not indicate completion of the sanitize operation). The completion of a sanitize operation is indicated in the Sanitize Status log page, and with either the Sanitize Operation Completed asynchronous event or the Sanitize Operation Completed With Unexpected Deallocation asynchronous event (if an Asynchronous Event Request Command is outstanding).

The Sanitize Capabilities (SANICAP) field of the Identify Controller data structure (refer to Figure 275) indicates the sanitize operation types supported and controller attributes specific to sanitize operations.

The sanitize operation types are:

- The Block Erase sanitize operation alters user data with a low-level block erase method that is specific to the media for all locations on the media within the NVM subsystem in which user data may be stored;
- The Crypto Erase sanitize operation alters user data by changing the media encryption keys for all locations on the media within the NVM subsystem in which user data may be stored; and
- The Overwrite sanitize operation alters user data by writing a fixed data pattern or related patterns to all locations on the media within the NVM subsystem in which user data may be stored one or more times. Figure 474 defines the data pattern or patterns that are written.

Controller attributes specific to sanitize operations include:

- The No-Deallocate Modifies Media After Sanitize (NODMMAS) field which indicates if media is modified by the controller after a sanitize operation successfully completes that had been requested with No-Deallocate After Sanitize set to '1' in the Sanitize command that started the sanitize operation; and
- No-Deallocate Inhibited (NDI) bit which indicates if the controller supports the No-Deallocate After Sanitize bit in the Sanitize Command.

The NODMMAS field in the Identify Controller data structure (refer to Figure 275), specifies that if a Sanitize command includes No-Deallocate After Sanitize set to '1' and NODMMAS is set to 10b, then a sanitize operation has an associated additional media modification operation. This additional media modification operation acts upon the results of the requested sanitize operation with the purpose of making all LBA contents readable. Refer to Annex A.3 for further information about sanitize operations and interactions with integrity circuits.

This additional media modification shall complete before the NVM subsystem:

a) reports sanitize completion by Asynchronous Event (refer to section 5.2); and
b) reports sanitize completion in the Sanitize Status log (refer to section 5.16.1.25).

The Overwrite sanitize operation is media specific and may not be appropriate for all media types. For example, if the media is NAND, multiple pass overwrite operations may have an adverse effect on media endurance.

Figure 474: Sanitize Operations – Overwrite Mechanism

OIPBP[1]	Overwrite Pass Count[1]	Overwrite Pass Number	User Data except PI Metadata	Protection Information[2]
'0'	All	All	Overwrite Pattern[1]	Each byte set to FFh
'1'	Even	First	Inversion of Overwrite Pattern[1]	Each byte cleared to 00h
		Subsequent	Inversion of Overwrite Pattern[1] from previous pass (i.e., each bit XORed with '1')	
'1'	Odd	First	Overwrite Pattern[1]	Each byte set to FFh
		Subsequent	Inversion of Overwrite Pattern[1] from previous pass (i.e., each bit XORed with '1')	

Notes:
1. Parameters are specified in Command Dword 10 and Command Dword 11 of the corresponding Sanitize command that started the Overwrite operation. The Overwrite Invert Pattern Between Passes (OIPBP) field is defined in Command Dword 10. The Overwrite Pass Count is defined in Command Dword 10. The Overwrite Pattern is defined in Command Dword 11. Refer to section 5.24.
2. If Protection Information is present within the metadata.

To start a sanitize operation, the host submits a Sanitize command specifying one of the sanitize operation types (i.e., Block Erase, Overwrite, or Crypto Erase). The host sets command parameters, including the Allow Unrestricted Sanitize Exit bit and the No-Deallocate After Sanitize bit. After validating the Sanitize command parameters, the controller starts the sanitize operation in the background, updates the Sanitize Status log page and then completes the Sanitize command with Successful Completion status. If the sanitize operation is to be followed by an associated additional media modification operation (refer to NODMMAS in Figure 275), then the associated additional media modification operation shall be completed before the controller reports sanitize operation complete. If a Sanitize command is completed with any status code other than Successful Completion, then the controller shall not start the sanitize operation and shall not update the Sanitize Status log page. The controller ignores Critical Warning(s) in the SMART / Health Information log page (e.g., read only mode) and attempts to complete the sanitize operation requested. Refer to section 5 for further information about restrictions on Admin Commands during the processing of a Format NVM command.

Following a successful sanitize operation, the values of user data, protection information, and non-PI metadata that result from an audit (refer to section 1.5.6) of the NVM subsystem are defined in the I/O command set specifications.
The Sanitize Status log page (refer to section 5.16.1.25) contains estimated times for sanitize operations and a consistent snapshot of information about the most recently started sanitize operation, including whether a sanitize operation is in progress, the sanitize operation parameters and the status of the most recent sanitize operation. The controller shall report sanitize operation in progress if either a sanitize operation is in progress or an associated additional media modification operation is in progress. If a sanitize operation is not in progress, then the Global Data Erased bit in the log page indicates whether the NVM subsystem may contain any user data (i.e., has not been written to since the most recent successful sanitize operation).

The Sanitize Status log page shall be updated as described:

- Initialize before any controller in the NVM subsystem is ready as described in sections 3.5.3 and 3.5.4;
- Update before a Sanitize command that starts a sanitize operation is completed (i.e., prior to the completion queue entry being posted for the Sanitize command); and
- Update when a sanitize operation is complete (e.g., immediately prior to the completion queue entry being posted for the Sanitize Operation Completed asynchronous event or for the Sanitize Operation Completed With Unexpected Deallocation asynchronous event).

The Sanitize Status log page should be updated periodically during a sanitize operation to make progress information available to hosts.

During a sanitize operation, the host may periodically examine the Sanitize Status log page to check for progress, however, the host should limit this polling (e.g., to at most once every several minutes) to avoid interfering with the progress of the sanitize operation itself.

On completion of a sanitize operation:

- If the sanitize operation is successful, then the Global Data Erased bit shall be set to ‘1’;
- The Sanitize Status log page is updated;
- The controller to which the Sanitize command was submitted completes an Asynchronous Event Request command (if one is outstanding) with the following information:
  - The Log Page Identifier field is set to 81h (i.e., Sanitize Status);
  - The Asynchronous Event Information field is set to Sanitize Operation Completed or to Sanitize Operation Completed With Unexpected Deallocation asynchronous event (refer to section 5.2); and
  - The Asynchronous Event Type field is set to 110b (i.e., I/O Command specific status);

  and
- All controllers in the NVM subsystem may resume any power management that was suspended when the sanitize operation started.

Upon completion of a sanitize operation, the host should read the Sanitize Status log page with the Retain Asynchronous Event bit cleared to ‘0’ (which clears the asynchronous event, if one was generated).

If a sanitize operation fails, all controllers in the NVM subsystem shall abort any command not allowed during a sanitize operation with a status code of Sanitize Failed (refer to section 8.21.1) until a subsequent sanitize operation is started or successful recovery from the failed sanitize operation occurs. A subsequent successful sanitize operation or the Exit Failure Mode action may be used to recover from a failed sanitize operation. Refer to section 5.24 for recovery details.

If the Sanitize command is supported, then the NVM subsystem and all controllers shall:

- Support the Sanitize Status log page;
- Support the Sanitize Operation Completed asynchronous event;

- Support the Sanitize Operation Completed With Unexpected Deallocation asynchronous event, if the Sanitize Config feature is supported;
- Support the Exit Failure Mode action for a Sanitize command;
- Support at least one of the following sanitize operation types: Block Erase, Overwrite, or Crypto Erase;
- Support the same set of sanitize operation types; and
- Indicate the supported sanitize operation types in the Sanitize Capabilities field in the Identify Controller data structure.

The Sanitize Config Feature Identifier (refer to section 5.27.1.19) contains the No-Deallocate Response Mode (NODRM) bit that specifies the response of the controller to a Sanitize command processed with the No-Deallocate After Sanitize bit (refer to Figure 303) set to '1' if the No-Deallocate Inhibited bit is set to '1' in the Sanitize Capabilities field of the Identify Controller data structure (refer to Figure 275). In the No-Deallocate Error Response Mode, the controller aborts such Sanitize commands with a status code of Invalid Field in Command. In the No-Deallocate Warning Response Mode, the controller processes such Sanitize commands, and if a resulting sanitize operation is completed successfully, then bits 2:0 of the Sanitize Status field are set to 100b in the Sanitize Status log page (refer to Figure 267).

8.21.1 Sanitize Operation Restrictions

While performing a sanitize operation and while a failed sanitize operation has occurred but successful recovery from that failure has not occurred, all enabled controllers and namespaces in the NVM subsystem are restricted to performing only a limited set of actions.

While a sanitize operation is in progress:

- All controllers in the NVM subsystem shall only process the Admin commands listed in Figure 140 subject to the additional restrictions stated in that figure;
- All I/O Commands other than a Flush command shall be aborted with a status code of Sanitize In Progress;
- Processing of a Flush command is specified in section 7.1;
- Any command or command option that is not explicitly permitted in Figure 140 shall be aborted with a status code of Sanitize In Progress if fetched by any controller in the NVM subsystem; and
- The Persistent Memory Region shall be prevented from being enabled (i.e., setting PMRCTL.EN to '1' does not result in PMRSTS.NRDY being cleared to '0').

While a failed sanitize operation has occurred, a subsequent sanitize operation has not started and successful recovery from the failed sanitize operation has not occurred:

- All controllers in the NVM subsystem shall only process the Sanitize command (refer to section 5.24) and the Admin commands listed in Figure 140 subject to the additional restrictions noted in that figure;
- All I/O Commands other than a Flush command (refer to section 7.1) shall be aborted with a status code of Sanitize Failed;
- The Sanitize command is permitted with action restrictions (refer to section 5.24);
- Aside from the Sanitize command, any other command or command option that is not explicitly permitted in Figure 140 shall be aborted with a status code of Sanitize Failed if fetched by any controller in the NVM subsystem; and
- The Persistent Memory Region shall be prevented from being enabled (i.e., setting PMRCTL.EN to '1' does not result in PMRSTS.NRDY being cleared to '0').

8.22 Submission Queue (SQ) Associations

When Predictable Latency Mode is enabled, all I/O commands for namespaces in a given NVM Set have the same quality of service attributes and shall exhibit predictable latencies as described in section 8.16.

The SQ Associations capability provides hints to the controller as to which specific I/O Queues are associated with a given NVM Set. The controller uses this information to further enhance performance when Predictable Latency Mode is enabled.

The SQ Associations capability is an optional capability. Predictable Latency Mode (refer to section 8.16) is not dependent on the use of the SQ Associations capability.

If a controller supports SQ Associations, then the controller shall:

- indicate support for the SQ Associations capability in the Controller Attributes (CTRATT) field in the Identify Controller data structure;
- indicate support for NVM Sets in the Controller Attributes (CTRATT) field in the Identify Controller data structure; and
- indicate support for Predictable Latency Mode in the Controller Attributes (CTRATT) field in the Identify Controller data structure (refer to Figure 275).

The host enables the SQ Associations capability by creating an association between an NVM Set and a Submission Queue at the time the Submission Queue is created (e.g., with a Create I/O Submission Queue command (refer to section 5.5)).

For the SQ Associations capability to yield benefits, the host is required to:

a) create an association between each Submission Queue and some NVM Set; and
b) only issue I/O commands to Submission Queues that have an association with the NVM Set that contains the namespace associated with the Namespace Identifier specified in that I/O command.

While this capability is enabled, failure to follow the specified operating rules may impact Predictable Latency (refer to section 8.16).

8.23 Standard Vendor Specific Command Format

Controllers may support the standard Vendor Specific command format defined in Figure 88. Host storage drivers may use the Number of Dwords fields to ensure that the application is not corrupting physical memory (e.g., overflowing a data buffer). The controller indicates support of this format in the Identify Controller data structure in Figure 275; refer to Admin Vendor Specific Command Configuration and NVM Vendor Specific Command Configuration.

8.24 Telemetry

Telemetry enables manufacturers to collect internal data logs to improve the functionality and reliability of products. The telemetry data collection may be initiated by the host or by the controller. The data is returned in the Telemetry Host-Initiated log page or the Telemetry Controller-Initiated log page (refer to section 5.16.1.8 and 5.16.1.9). The data captured is vendor specific. The telemetry feature defines the mechanism to collect the vendor specific data. The controller indicates support for the telemetry log pages and for the Data Area 4 size in the Log Page Attributes (LPA) field in the Identify Controller data structure (refer to Figure 275).

An important aspect to discovering issues by collecting telemetry data is the ability to qualify distinct issues that are being collected. The ability to create a one to one mapping of issues to data collections is essential. If a one to one mapping is not established, there is the risk that several payload collections appear distinct but are actually all caused by the same issue. Conversely, a single payload collection may have payloads caused by several issues mixed together creating additional complexity in determining the root cause. As a result, flexibility in size is provided in the collection of telemetry payloads and a three phase process is typically used.

The first phase establishes that an issue exists and is best accomplished by collecting a minimum set of data to identify the issue as being distinct from other issues. Once the number of instances of an issue establish an investigation, another phase may be necessary to collect actionable information. In the second phase, a targeted collection of more in depth medium size payloads are gathered and analyzed to identify the source of the problem.

If the small or medium sized telemetry data collection provides insufficient information, a third phase may be employed to collect additional details. If bit 6 is cleared to ‘0’ in the Log Page Attributes field, then the third phase provides the largest and most complete payload to diagnose the issue. If bit 6 is set to ‘1’ in the Log Page Attribute and the Extended Telemetry Data Area 4 Supported (ETDAS) field is set to 1h in the

Host Behavior Support feature (refer to section 5.27.1.18) then a fourth phase may be employed to collect the largest and most complete payload to diagnose the issue. If Data Area 4 is created, then Data Area 3 of non-zero length shall also be created and populated as part of data collection.

There are two telemetry data logs (i.e., Telemetry Host-Initiated log page and Telemetry Controller-Initiated log page) defined. Each telemetry data log is made up of a single set of Telemetry Data Blocks. Each Telemetry Data Block is 512 bytes in size. Telemetry data is returned (refer to section 5.16.1.8 and section 5.16.1.9) in units of Telemetry Data Blocks. Each telemetry data log is segmented into:

a) Three Telemetry Data Areas (i.e., small, medium, and large), if bit 6 of the Log Page Attributes field is cleared to '0'; or
b) Four Telemetry Data Areas (i.e., small, medium, large and extra-large) If bit 6 of the Log Page Attributes field is set to '1' and the Extended Telemetry Data Area 4 Supported (ETDAS) field is set to 1h in the Host Behavior Support feature (refer to section 5.27.1.18).

All telemetry data areas start at Telemetry Data Block 1.

Each Telemetry Data Area shall represent the controller's internal state at the time the telemetry data was captured.

Each Telemetry Data Area is intended to capture a richer set of data to aid in resolution of issues. Telemetry Data Area 1 is intended to have a small size payload (i.e., the first phase), Telemetry Data Area 2 is intended to have a medium size payload (i.e., the second phase), and Telemetry Data Area 3 is intended to have a large size payload (i.e., the third phase). Telemetry Data Area 4 is intended to have an extra-large size payload (i.e. the fourth phase). The size of each Telemetry Data Area is vendor specific and may change on each data collection. When possible, the host should retrieve the payload for all supported Telemetry Data Areas to enable the best diagnosis of the issue(s).

The preparation, collection, and submission of telemetry data is similar for host-initiated and controller-initiated data; the primary difference is the trigger for the collection. The operational model for telemetry is:

1. The host identifies controller support for Telemetry log pages in the Identify Controller data structure;
2. The host may indicate the support for the Telemetry Host-Initiated Data Area 4 and Telemetry Controller-Initiated Data Area 4 by setting the Extended Telemetry Data Area 4 Supported (ETDAS) field to 1h in the Host Behavior Support feature (refer to section 5.27.1.18);
3. The host prepares an area to store telemetry data if needed;
4. To receive notification that controller-initiated telemetry data is available, the host enables Telemetry Log Notices using the Asynchronous Event Configuration feature (refer to section 5.27.1.8); and
5. If the host decides to collect host-initiated telemetry data or the controller signals that controller-initiated telemetry data is available:

    a. The host reads the appropriate blocks of the Telemetry Data Area from the Telemetry Host-Initiated log page (refer to section 5.16.1.8) or the Telemetry Controller-Initiated log page (refer to section 5.16.1.9). If possible, the host should collect Telemetry Data Area 1, 2, 3, and 4. The host reads the log in 512 byte Telemetry Data Block units (i.e., a starting offset that is a multiple of 512, and a length that is a multiple of 512). The host should set the Retain Asynchronous Event bit to '1';
    b. The host re-reads the header of the log page and ensures that the Telemetry Host-Initiated Data Generation Number field from the Telemetry Host-Initiated log page or the Telemetry Controller-Initiated Data Generation Number field in the Telemetry Controller-Initiated log page matches the original value read. If these values do not match, then the data captured is not consistent and should be re-read from the log page with the Retain Asynchronous Event bit set to '1';
    c. If the host is reading the Telemetry Controller-Initiated log page, then the host reads any portion of that log page with the Retain Asynchronous Event bit cleared to '0' to indicate to the controller that the host has completed reading the Telemetry Controller-Initiated log page; and

d. When all telemetry data has been saved, the data should be forwarded to the manufacturer of the controller.

The trigger for the collection for host-initiated data is typically a system crash, but may also be initiated during normal operation. The host proceeds with a host-initiated data collection by submitting the Get Log Page command for the Telemetry Host-Initiated log page with the Create Telemetry Host-Initiated Data bit set to '1' in the Log Specific Parameter field. The controller should complete the command quickly (e.g., in less than one second) to avoid a user rebooting the system prior to completion of the data collection.

The NVM subsystem is allowed to provide a Telemetry Host-Initiated log page per controller or a shared Telemetry Host-Initiated log page across all controllers in the NVM subsystem. If a shared Telemetry Host-Initiated log page is implemented, the Telemetry Host-Initiated Data Generation Number field in the Telemetry Host-Initiated log page is used to allow the host to detect that the Telemetry Host-Initiated log page has been changed by a host through a different controller.

The controller notifies the host to collect controller-initiated data through the completion of an Asynchronous Event Request command with an Asynchronous Event Type of Notice that indicates a Telemetry Log Changed event. The host may also determine controller-initiated data is available via the Telemetry Controller-Initiated Data Available field in the Telemetry Host-Initiated or the Telemetry Controller-Initiated log pages. The host proceeds with a controller-initiated data collection by submitting the Get Log Page command for the Telemetry Controller-Initiated log page. Once the host has started reading the Telemetry Controller-Initiated log page, the controller should avoid modifying the controller-initiated data until the host has finished reading all controller-initiated data. The amount of time for the host to read the controller-initiated data is vendor specific.

Since there is only one set of controller-initiated data, the controller is responsible for prioritizing the version of the controller-initiated data that is available for the host to collect. When the controller replaces the controller-initiated data with new controller-initiated data, the controller shall increment the Telemetry Controller-Initiated Data Generation Number field. The host needs to ensure that the Telemetry Controller-Initiated Data Generation Number field has not changed between the start and completion of the controller-initiated data collection to ensure the data captured is consistent.

8.24.1 Telemetry Data Collection Examples (Informative)

This section includes several examples of Telemetry Host-Initiated Data Areas for illustration. The same concepts apply to the Telemetry Controller-Initiated Data Areas.

If a Telemetry Host-Initiated log page has no data for collection, then the following fields are all cleared to 0h:

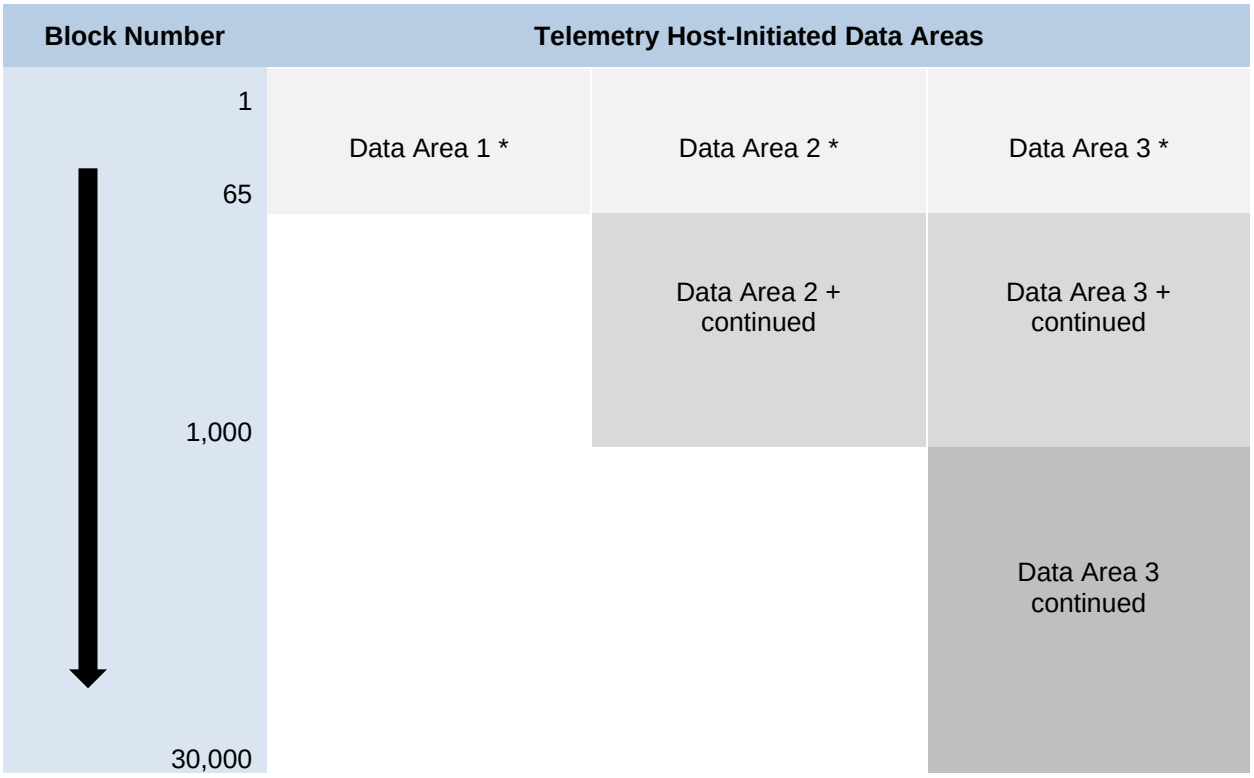
- Telemetry Host-Initiated Data Area 1 Last Block = 0;
- Telemetry Host-Initiated Data Area 2 Last Block = 0; and
- Telemetry Host-Initiated Data Area 3 Last Block = 0.

When all three telemetry data areas are populated, then the Telemetry Host-Initiated log page has different values in each of the Telemetry Host-Initiated Data Area n Last Block fields. For example, the following values correspond to the layout shown in Figure 475:

- Telemetry Host-Initiated Data Area 1 Last Block = 65;
- Telemetry Host-Initiated Data Area 2 Last Block = 1,000; and
- Telemetry Host-Initiated Data Area 3 Last Block = 30,000.

As a result of telemetry data areas being made up of a single set of Telemetry Data Blocks starting at Telemetry Data Block 1, the telemetry data contained in Telemetry Data Block 1 through Telemetry Data Block 65 of data area 1, data area 2, and data area 3 is the same. In addition, the telemetry data contained in Telemetry Data Block 66 through Telemetry Data Block 1,000 of data area 2 and data area 3 is the same.

Figure 475: Telemetry Log Example – All Data Areas Populated



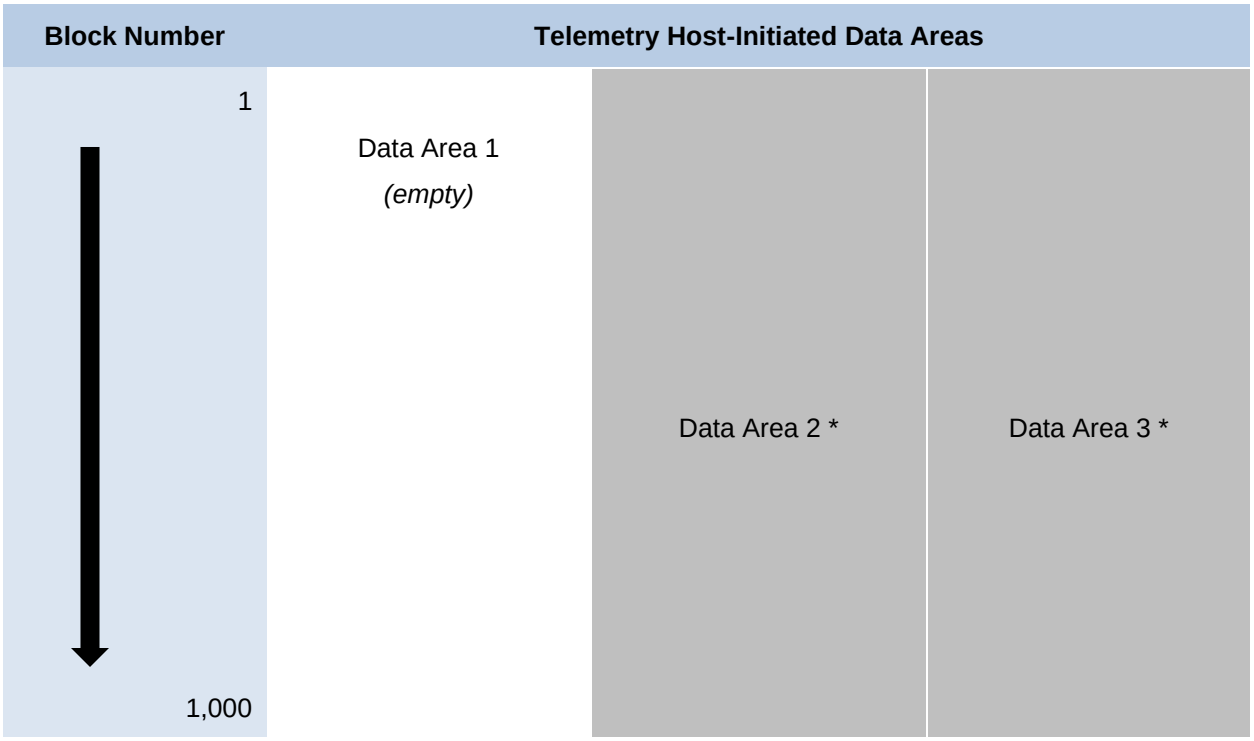
* Data Area 1, Data Area 2, and Data Area 3 contain the same telemetry data in blocks 1 through 65.
\+ Data Area 2 and Data Area 3 contain the same telemetry data in blocks 66 through 1,000.

When only the second data areas is populated, then the Telemetry Host-Initiated log page has no data in Telemetry Data Area 1 shown by having its corresponding last block value cleared to 0h, and no additional data in Telemetry Data Area 3 shown by having its corresponding last block value set to the same value as the last block value for Telemetry Data Area 2. For example, the following values correspond to the layout shown in Figure 476:

- Telemetry Host-Initiated Data Area 1 Last Block = 0;
- Telemetry Host-Initiated Data Area 2 Last Block = 1,000; and
- Telemetry Host-Initiated Data Area 3 Last Block = 1,000.

As a result of telemetry data areas being made up of a single set of Telemetry Data Blocks starting at Telemetry Data Block 1, the telemetry data contained in Telemetry Data Block 1 through Telemetry Data Block 1,000 of data area of data area 2 and data area 3 is the same.

Figure 476: Telemetry Log Example – Data Area 2 Populated



* Data Area 2, and Data Area 3 contain the same telemetry data in blocks 1 through 1,000.

8.25 Universally Unique Identifiers (UUIDs) for Vendor Specific Information

8.25.1 UUIDs for Vendor Specific Information Introduction

Several commands send or receive information that contains fields described as Vendor Specific or that is specified by a command field containing a value in a vendor specific range. Examples include the Set Features command, which may specify a vendor specific feature identifier, and the Identify command, which may retrieve a data structure having a vendor specific area.

The vendor specific information may have different definitions (e.g., a vendor specific log page identifier with the contents of the page defined differently by different entities, such as an NVM subsystem vendor and an NVM subsystem customer). By associating each definition of the information with a UUID specified by the defining entity, a command is able to specify the particular definition of the information.

A command specifies a particular definition of the information by specifying an index into a list of UUIDs supported by the controller (refer to section 5.17.2.16). The NVMe Invalid UUID (refer to section 8.25.2) is used to replace a previously valid UUID in the UUID List (refer to Figure 284). This is done to keep the values in the list at a static index, as that index is used by the Host to access the UUID List contents.

NVM subsystem vendors and customers communicate (by means outside the scope of this specification) the UUID used for each definition of the information.

8.25.2 UUIDs for Vendor Specific Information Requirements

A UUID list is a list of non-zero UUID values, terminated by a 0h UUID value. Each non-zero UUID value may be either a valid UUID or the NVMe Invalid UUID. The NVMe Invalid UUID is the hexadecimal value FFFFFFFF_FFFFFFFF_7FFFFFFF_FFFFFFFFh. A valid UUID is any non-zero value other than the NVMe Invalid UUID.

If a command supports selection of a UUID, then the UUID Selection Supported bit in the Commands Supported and Effects data structure for that command (refer to Figure 211) shall be set to '1'. If a command does not support selection of a UUID, then the UUID Selection Supported bit shall be cleared to '0'.

If the UUID Selection Supported bit is set to '1' for one or more commands, then the UUID List bit in the Controller Attributes field shall be set to '1' (refer to Figure 275), and the controller shall support reporting of a UUID List (refer to Figure 284).

If a command supports selection of a UUID, then that command contains a UUID Index field (refer to Figure 477).

Figure 477: UUID Index Field

Bits	Description
6:0	**UUID Index:** If this field is set to a non-zero value, then the value of this field is the index of a UUID in the UUID List (refer to Figure 284) that is used by the command. If this field is cleared to 0h, then no UUID index is specified.

If the UUID Index field specifies a valid UUID (i.e., the UUID Index field is set to a non-zero value and the UUID at that index indicates a valid UUID) (refer to section 5.17.2.16), then the controller shall process the command using the vendor specific information specified by that UUID. If the UUID Index field is cleared to 0h, then the command does not specify a UUID.

If no UUID is specified by the command, then the controller shall process the command, returning vendor specific information.

The controller shall abort the command with a status code of Invalid Field in Command if:

a) The controller does not support the UUID specified by the UUID Index for the specified information;
b) The UUID specified by the UUID Index is cleared to 0h; or
c) The UUID specified by the UUID Index is the NVMe Invalid UUID.

If a firmware image is activated that has a UUID List in which an entry is different from that of the previously-active firmware image, then a host that is unaware of the change may issue a command with the UUID index value for that entry. Such a command may produce unexpected results because the UUID specified by that UUID Index has changed. To avoid this, vendors should follow the following revision guidelines for UUID lists when constructing firmware images that support UUID selection:

a) Add UUIDs that are not supported in prior firmware image revisions to the end of the UUID List in subsequent firmware image revisions;
b) Remove UUIDs that are supported in prior firmware image revisions by replacing the UUID with the NVMe Invalid UUID in the same entry in the UUID list in subsequent firmware image revisions;
c) Do not replace the NVMe Invalid UUID with a valid UUID in the same UUID list entry in subsequent firmware image revisions; and
d) Do not shorten or remove the UUID list in subsequent firmware image revisions.

In these guidelines, the terms "prior" and "subsequent" refer to a linear sequence of firmware versions (e.g., based on the date and time of the construction of the downloadable firmware image).

Following these guidelines prevents the host from inadvertently specifying the wrong UUID because there is at most one valid UUID for each entry in the UUID list. Hence a command that specifies a UUID Index either specifies the intended UUID or is aborted because that entry in the UUID list is empty or contains the NVMe Invalid UUID.

The controller shall require a reset to activate a downloaded firmware image (refer to section 5.12) if the downloaded image reports a UUID list with at least one slot in which a valid UUID replaces the NVMe Invalid UUID or a different valid UUID in the existing image. All controllers that are affected by the UUID list change caused by activation of a downloaded firmware image shall be reset as part of activating that downloaded firmware image.

The above requirements for a reset to activate a downloaded firmware image do not require the controller to directly compare the UUID lists in the current and downloaded firmware images. For example, a vendor

could use a vendor-specific major.minor firmware image revision numbering system (e.g., 3.5, 4.1) where all downloadable firmware images with the same major revision number follow the above guidelines. In that scenario, the controller is able to meet these reset requirements by requiring a reset if the downloaded firmware image and the currently executing firmware have different major revision numbers.

8.25.3 UUIDs for Vendor Specific Information Examples

This section includes examples of the use of UUIDs to select vendor specific information.

8.25.3.1 Vendor Specific Log Page Example

If entity C and entity V create different definitions for a vendor specific log page having the same log page identifier (e.g., D0h), then each assigns a UUID to distinguish their definition (e.g., entity V assigns UUID V and entity C assigns UUID C).

A controller supporting both definitions of the log page:

a) Sets the UUID List bit to ‘1’ in the CTRATT field of the Identify Controller data structure (refer to Figure 275);
b) Sets the UUID Selection Supported bit to ‘1’ in the Commands Supported and Effects data structure (refer to Figure 211) corresponding to the Get Log Page command; and
c) Reports both UUID V and UUID C in the UUID list (refer to Figure 284).

A host requesting the log page defined by entity C:

1) Determines the index of UUID C in the UUID list;
2) Sets the Log Page Identifier field of the Get Log Page command to D0h; and
3) Sets the UUID Index field of the Get Log Page command to the index of UUID C.

A host requesting the log page defined by entity V:

1) Determines the index of UUID V in the UUID list;
2) Sets the Log Page Identifier field of the Get Log Page command to D0h; and
3) Sets the UUID Index field of the Get Log Page command to the index of UUID V.

A host not specifying the definition of the log page clears the UUID Index field to 0h. The selection of the log page definition returned by the controller is vendor specific (e.g., the controller may select any definition for the returned data).

8.25.3.2 Vendor Specific Feature Example

If entity C and entity V create different definitions for a vendor specific feature having the same Feature Identifier (e.g., F1h), then each assigns a UUID to distinguish their definitions (e.g., entity V assigns UUID V and entity C assigns UUID C).

A controller supporting both definitions of the feature for the Get Features command:

a) Sets the UUID List bit to ‘1’ in the CTRATT field of the Identify Controller data structure (refer to Figure 275);
b) Sets the UUID Selection Supported bit to ‘1’ in the Commands Supported and Effects data structure (refer to Figure 211) corresponding to the Get Features command;
c) Sets the UUID Selection Supported bit to ‘1’ in the FID Supported and Effects log page (refer to Figure 256); and
d) Reports both UUID V and UUID C in the UUID list (refer to Figure 284).

A host retrieving the attributes of the feature defined by entity C:

1) Determines the index of UUID C in the UUID list;
2) Sets the Feature Identifier field of the Get Features command to F1h; and
3) Sets the UUID Index field of the Get Features command to the index of UUID C.

A host retrieving the attributes of the feature defined by entity V:

1) Determines the index of UUID V in the UUID list;

2) Sets the Feature Identifier field of the Get Features command to F1h; and
3) Sets the UUID Index field of the Get Features command to the index of UUID V.

8.26 Virtualization Enhancements

Virtualized environments may use an NVM subsystem with multiple controllers to provide virtual or physical hosts direct I/O access. The NVM subsystem is composed of primary controller(s) and secondary controller(s), where the secondary controller(s) depend on primary controller(s) for dynamically assigned resources. A host may issue the Identify command to a primary controller specifying the Secondary Controller List to discover the secondary controllers associated with that primary controller. All secondary controllers shall be part of the same domain as the primary controller with which they are associated.

Controller resources may be assigned or removed from a controller using the Virtualization Management command issued to a primary controller. The following types of controller resources are defined:

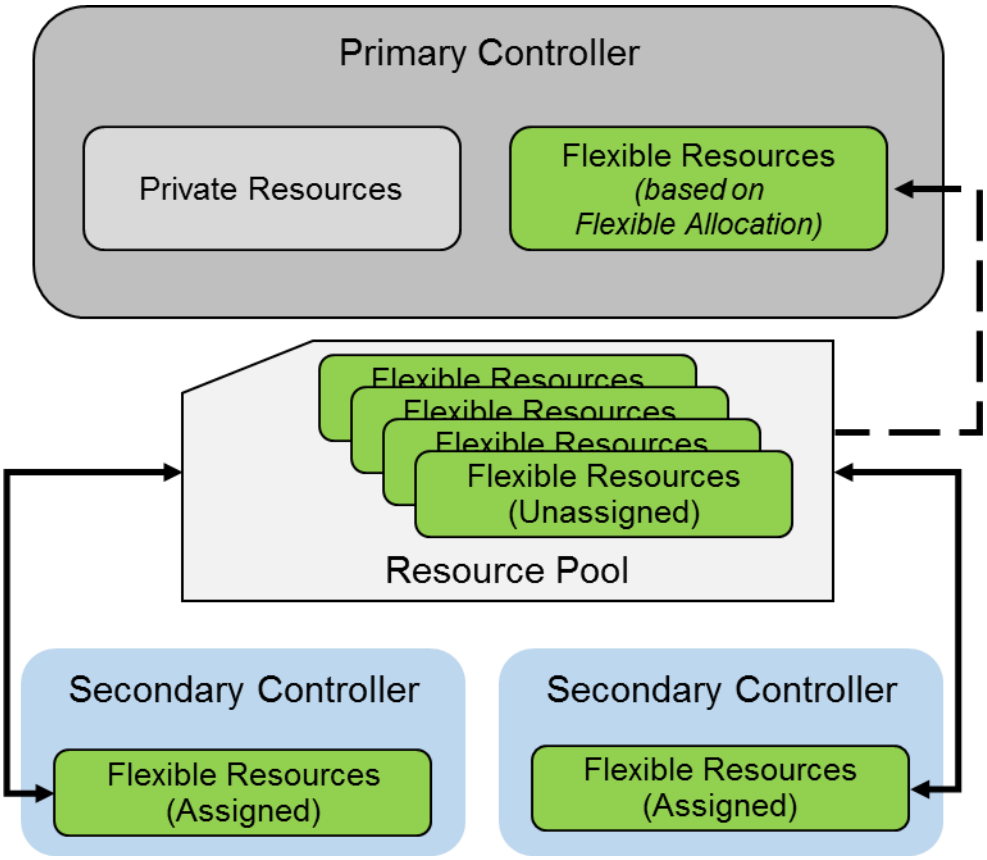
- Virtual Queue Resource (VQ Resource): a type of controller resource that manages one Submission Queue (SQ) and one Completion Queue (CQ) (refer to section 8.26.1); and
- Virtual Interrupt Resource (VI Resource): a type of controller resource that manages one interrupt vector (refer to section 8.26.2).

Flexible Resources are controller resources that may be assigned to the primary controller or one of its secondary controllers. The Virtualization Management command is used to provision the Flexible Resources between a primary controller and one of its secondary controller(s). A primary controller's allocation of Flexible Resources may be modified using the Virtualization Management command and the change takes effect after any Controller Level Reset other than a Controller Reset (i.e., CC.EN transitions from '1' to '0'). A secondary controller only supports having Flexible Resources assigned or removed when in the Offline state.

Private Resources are controller resources that are permanently assigned to a primary or secondary controller. These resources are not supported by the Virtualization Management command.

The primary controller is allowed to have a mix of Private and Flexible Resources for a particular controller resource type. If there is a mix, then the Private Resources occupy the lower contiguous range of resource identifiers starting with 0. Secondary controllers shall have all Private or all Flexible Resources for a particular resource type. Controller resources assigned to a secondary controller always occupy a contiguous range of identifiers with no gaps, starting with 0. If a particular controller resource type is supported as indicated in the Controller Resource Types field of the Primary Controller Capabilities Structure, then all secondary controllers shall have that controller resource type assigned as a Flexible Resource. Figure 478 shows the controller resource allocation model for a controller resource type that is assignable as a Flexible Resource.

Figure 478: Controller Resource Allocation



For each controller resource type supported, the Primary Controller Capabilities Structure (refer to Figure 281) defines:

- The total number of Flexible Resources;
- The total number of Private Resources for the primary controller;
- The maximum number of Flexible Resources that may be assigned to a secondary controller using the Virtualization Management command; and
- The assignment of resources to the primary controller.

Primary and secondary controllers may implement all features of this specification, except where commands are defined as being only supported by a primary controller. It is recommended that only primary controllers support the privileged actions described in section 3.10 so that untrusted hosts using secondary controllers do not impact the entire NVM subsystem state.

The Secondary Controller List structure returned by the Identify command is used to determine the topology of secondary controllers and the resources assigned. The secondary controller shall be in the Offline state to configure resources. The Virtualization Management command is used to transition the secondary controller between the Online state and the Offline state. Refer to section 8.26.3 for details on the Online and Offline states.

To support the Virtualization Enhancements capability, the NVM subsystem shall support the following:

- One or more primary controllers, each of which supports:
  - One or more secondary controllers;
  - A pool of unassigned Flexible Resources that supports allocation to a primary controller and dynamic assignment to its associated secondary controllers;

  - Two or more Private Resource queue pairs;
  - Indicate support for the Virtualization Management command in the Optional Admin Command Support (OACS) field in the Identify Controller data structure;
  - The Virtualization Management command;
  - The Primary Controller Capabilities Structure defined in Figure 281 (Identify command with CNS value of 14h);
  - The Secondary Controller List defined in Figure 282 (Identify command with CNS value of 15h); and
  - The Namespace Management capability (refer to section 8.11);
- One or more secondary controllers; and
- Flexible Resources, each of which supports all of the following:
  - Assignment and removal by exactly one primary controller; and
  - Assignment to no more than one controller at a time.

Within an NVM subsystem that supports both the Virtualization Enhancements capability and SR-IOV (refer to section 8.26.4), all controllers that are SR-IOV PFs shall be primary controllers, and all controllers that are SR-IOV VFs shall be secondary controllers of their associated PFs.

8.26.1 VQ Resource Definition

A Virtual Queue Resource (VQ Resource) is a type of controller resource that manages one CQ and one SQ. For a VQ Resource that is assigned to a controller, its resource identifier is equivalent to its Queue Identifier.

The Controller Resource Types field of the Primary Controller Capabilities Structure indicates whether VQ Resources are supported. If VQ Resources are unsupported, a primary controller and its associated secondary controllers have all queues as Private Resources. The rest of this section assumes that VQ Resources are supported.

The secondary controller is assigned VQ Resources using the Virtualization Management command. The number of VQ Resources assigned is discoverable in the Secondary Controller List entry for the associated secondary controller. The number of VQ Resources assigned may also be discovered using the Get Features command with the Number of Queues Feature identifier (refer to section 5.27.1.5).

If a secondary controller has no assigned VQ Resources, then that controller remains in the Offline state. A secondary controller is not able to transition to the Online state until VQ Resources for an Admin Queue and one or more I/O Queues have been assigned to that controller (i.e., the minimum number of VQ Resources that may be assigned is two).

A primary controller that supports VQ Resources shall have at least two queue pairs that are Private Resources to ensure there is a minimum of an Admin Queue pair and one I/O queue pair for the primary controller at all times. A primary controller may be allocated VQ Resources using the Primary Controller Flexible Allocation action of the Virtualization Management command. The VQ resources allocated take effect after a Controller Level Reset and are persistent across power cycles and resets. The number of VQ Resources currently allocated is discoverable in the Primary Controller Capabilities Structure. The number of VQ Resources currently allocated may also be discovered using the Get Features command with the Number of Queues Feature identifier (refer to section 5.27.1.5).

8.26.2 VI Resource Definition

A Virtual Interrupt Resource (VI Resource) is a type of controller resource that manages one interrupt vector, such as an MSI-X vector. For a VI Resource that is assigned to a controller, its resource identifier is equivalent to its interrupt vector number.

The Controller Resource Types field of the Primary Controller Capabilities Structure indicates whether VI Resources are supported. If VI Resources are unsupported, a primary controller and its associated secondary controllers have all interrupts as Private Resources. The rest of this section assumes that VI Resources are supported.

The secondary controller is assigned VI Resources using the Virtualization Management command. The number of VI Resources assigned is discoverable in the Secondary Controller List entry for the associated secondary controller.

While a primary controller and/or its associated secondary controllers may concurrently support multiple types of interrupt vectors (e.g., MSI and MSI-X), all the controllers' VI Resources shall contain interrupt resources for interrupt vectors of the same type. In this revision, MSI-X is the only supported type of VI Resource.

For a secondary controller that supports VI Resources with MSI-X vectors, if at least one VI Resource is assigned to that controller, MSIXCAP.MXC.TS (refer to the MSI-X Capability section of the NVMe over PCIe Transport Specification) indicates the number of VI Resources assigned to the controller. Since MSIXCAP.MXC.TS is read-only, the value shall only be updated when the secondary controller is in the Offline state. MSI-X Table Entries on the secondary controller for newly assigned VI Resources shall be reset to default values.

If a secondary controller that supports VI Resources has no assigned VI Resources, then that controller remains in the Offline state. A secondary controller is not able to transition to the Online state until a VI Resource for interrupt vector 0 has been assigned to that controller. For a secondary controller that supports VI Resources with MSI-X vectors, if no VI Resources are assigned to that controller, then MSIXCAP.MXC.TS is reserved.

A primary controller that supports VI Resources shall have at least one interrupt that is a Private Resource. Interrupt vector 0 is always assigned to the primary controller. A primary controller may be allocated VI Resources using the Primary Controller Flexible Allocation action of the Virtualization Management command. The VI resources allocated take effect after a Controller Level Reset and are persistent across power cycles and resets. The number of VI Resources currently allocated is discoverable in the Primary Controller Capabilities Structure. For a primary controller that supports VI Resources with MSI-X vectors, MSIXCAP.MXC.TS indicates an MSI-X Table size equal to the total number of Private Resources and the Flexible Resources currently allocated following a Controller Level Reset.

When an I/O CQ is created, the controller supports mapping that I/O CQ to any valid interrupt vector, regardless of whether they have the same resource identifier, as long as the I/O CQ and the interrupt vector are attached to the same controller.

8.26.3 Secondary Controller States and Resource Configuration

A secondary controller shall be in one of the following states:

- **Online:** The secondary controller may be in use by a host. Required resources have been assigned. The secondary controller may be enabled in this state (CC.EN may be set to '1' and CSTS.RDY may then transition to '1'); or
- **Offline:** The secondary controller may not be used by a host. CSTS.CFS shall be set to '1'. Controller properties other than CSTS are undefined in this state.

The host may request a transition to the Online or Offline state using the Virtualization Management command. When a secondary controller transitions from the Online state to the Offline state all Flexible Resources are removed from the secondary controller.

To ensure that the host accurately detects capabilities of the secondary controller, the host should complete the following procedure to bring a secondary controller Online:

1. Use the Virtualization Management command to set the secondary controller to the Offline state;
2. Use the Virtualization Management command to assign VQ resources and VI resources;
3. Perform a Controller Level Reset. If the secondary controller is a VF, then this should be a VF Function Level Reset; and
4. Use the Virtualization Management command to set the secondary controller to the Online state.

If VI Resources are supported, then following this process ensures the MSI-X Table size indicated by MSIXCAP.MXC.TS is updated to reflect the appropriate number of VI Resources before the transition to the Online state.

A primary controller or secondary controller is enabled when CC.EN and CSTS.RDY are both set to ‘1’ for that controller. A secondary controller is able to be enabled only when in the Online state. If the primary controller associated with a secondary controller is disabled or undergoes a Controller Level Reset, then the secondary controller shall implicitly transition to the Offline state. A secondary controller shall transition to the Offline state when a shutdown occurs (refer to section 3.1.3.5 and section 3.1.3.20) on the primary controller associated with that secondary controller.

Resources shall only be assigned to a secondary controller when in the Offline state. If the minimum number of resources are not assigned to a secondary controller, then a request to transition to the Online state shall fail for that secondary controller. For implementations that support SR-IOV, if VF Enable is cleared to ‘0’ or NumVFs specifies a value that does not enable the associated secondary controller, then the secondary controller shall implicitly transition to the Offline state.

8.26.4 Single Root I/O Virtualization and Sharing (SR-IOV)

The PCI-SIG® Single Root I/O Virtualization and Sharing Specification (SR-IOV) defines extensions to PCI Express that allow multiple System Images (SIs), such as virtual machines running on a hypervisor, to share PCI hardware resources. The primary benefit of SR-IOV is that it eliminates the hypervisor from participating in I/O operations which may be a significant factor limiting storage performance in some virtualized environments and allows direct SI access to PCI hardware resources.

A Physical Function (PF) is a PCI Express Function that supports the SR-IOV Capability, which in turn allows that PF to support one or more dependent Virtual Functions (VFs). These PFs and VFs may support NVM Express controllers that share an underlying NVM subsystem with multi-path I/O and namespace sharing capabilities (refer to section 2.4.1).

SR-IOV Virtual Functions (VFs) with an NVM Express Class Code (refer to the PCI Header section of the NVMe over PCIe Transport Specification) shall implement fully compliant NVM Express controllers. This ensures that the same host software developed for non-virtualized environments is capable of running unmodified within an SI.

For hosts where SR-IOV is unsupported or not needed, a controller that is a PF shall support operation as a stand-alone controller.

For a controller that is a PF, the requirements for SR-IOV Capability registers VF BAR0, VF BAR1, VF BAR2, VF BAR4, and VF BAR5 are the same as the requirements for PCI registers BAR0, BAR1, BAR4, and BAR5, respectively. For a controller that is a PF, SR-IOV Capability register VF BAR2 shall not support Index/Data Pair. Refer to the PCI Header section of the NVMe over PCIe Transport Specification.

To accommodate SR-IOV address range isolation requirements, VF BAR2 and VF BAR3 may support a 64-bit prefetchable memory register space which shall only be used for MSI-X Tables and MSI-X PBAs of VFs. MSI-X Table BIR = ‘2’ and MSI-X PBA BIR = ‘2’ are valid for controllers that are VFs. Refer to the MSI-X Capability section of the NVMe over PCIe Transport Specification.

While the controller properties of a controller that is a VF are accessible only if SR-IOV Control.VF MSE is set to ‘1’, clearing VF MSE from ‘1’ to ‘0’ does not cause a reset of that controller. In this case, controller properties are hidden, but their values are not reset.

9 Error Reporting and Recovery

9.1 Command and Queue Error Handling

In the case of serious error conditions, like Completion Queue Invalid, the operation of the associated Submission Queue or Completion Queue may be compromised. In this case, host software should delete the associated Completion Queue and/or Submission Queue. The delete of a Submission Queue aborts all outstanding commands, and deletion of either queue type releases resources associated with that queue. Host software should recreate the Completion Queue and/or Submission Queue to then continue with operation.

In the case of serious error conditions for Admin commands, the entire controller should be reset using a Controller Level Reset. The entire controller should also be reset if a completion is not received for the deletion of a Submission Queue or Completion Queue.

For most command errors, there is not an issue with the Submission Queue and/or Completion Queue itself. Thus, host software and the controller should continue to process commands. It is at the discretion of host software whether to retry the failed command; the Retry bit in the completion queue entry indicates whether a retry of the failed command may succeed.

9.2 Media and Data Error Handling

In the event that the requested operation could not be performed to the NVM media, the particular command is completed with a media error indicating the type of failure using the appropriate status code.

If a read error occurs during the processing of a command, (e.g., End-to-end Guard Check Error, Unrecovered Read Error), the controller may either stop the DMA transfer into the memory or transfer the erroneous data to the memory. The host shall ignore the data in the memory locations for commands that complete with such error conditions.

If a write error occurs during the processing of a command, (e.g., an internal error, End-to-end Guard Check Error, End-to-end Application Tag Check Error), the controller may either stop or complete the DMA transfer.

Additional I/O Command Set specific error handling is described within applicable I/O Command Set specifications.

9.3 Memory Error Handling

For PCI Express implementations, memory errors such as target abort, master abort, and parity may cause the controller to stop processing the currently executing command. These are serious errors that cannot be recovered from without host software intervention.

A master/target abort error occurs when host software has provided, to the controller, the address of memory that does not exist. When this occurs, the controller aborts the command with a Data Transfer Error status code.

9.4 Internal Controller Error Handling

Errors such as a DRAM failure or power loss notification indicate that a controller level failure has occurred during the processing of a command. The status code of the completion queue entry should indicate an Internal Error status code. Host software shall ignore any data transfer associated with the command. The host may choose to re-submit the command or indicate an error to the higher level software.

9.5 Controller Fatal Status Condition

If the controller has a serious error condition and is unable to communicate with host software via completion queue entries in the Admin Completion Queue or I/O Completion Queues, then the controller may set the Controller Fatal Status (CSTS.CFS) bit to ‘1’ (refer to section 3.1.3.6). This indicates to host

software that a serious error condition has occurred. When this condition occurs, host software should attempt to reset and then re-initialize the controller.

The Controller Fatal Status condition is not indicated with an interrupt. If host software experiences timeout conditions and/or repeated errors, then host software should consult the Controller Fatal Status (CSTS.CFS) bit to determine if a more serious error has occurred.

If the Controller Fatal Status (CSTS.CFS) bit is set to '1' on any controller in the NVM subsystem, the host should issue a Controller Reset to that controller.

If that Controller Reset does not clear the Controller Fatal Status condition, the host should initiate an NVM Subsystem Reset (refer to section 3.7.1), if supported.

Performing an NVM Subsystem Reset (NSSR) may cause PCI Express links to go down as part of resetting the NVM subsystem. Host software may have undesirable effects related to PCI Express links going down (e.g., some host operating systems or hypervisors may crash).

NVM Subsystem Reset should not be used if the host software has undesirable effects related to PCI Express links going down. This host software includes, but is not limited to, operating systems using Firmware First Error Handling (refer to the ACPI Specification). Such operating systems should not use NSSR for recovery from CFS conditions.

Annex A. Sanitize Operation Considerations (Informative)

A.1 Overview

The Sanitize command initiates a sanitize operation that makes all user data previously written to the device inaccessible. To do this a Sanitize command is provided over the device's physical interface that cause the controller to process the requested operation. The actual result of the operation is very difficult to prove as complete. This annex provides some context and considerations for understanding the result of the operation and the practical limitations for auditing the result of the sanitize operation.

A.2 Hidden Storage (Overprovisioning)

Sanitize operations affect all physical storage that is able to hold user data. Many NVMe SSDs contain more physical storage than is addressable through the interface (overprovisioning), which is used for vendor specific purposes that may include providing increasing endurance, improving performance, and providing extra capacity to allow retiring bad or worn-out storage without affecting capacity. This excess capacity as well as any retired storage are not accessible through the interface. Vendor specific innovative use of this extra capacity supports advantages to the end user, but the lack of observability makes it difficult to ensure that all storage within the device has been affected. Only the accessible storage is able to be audited for the results of a sanitization operation.

A.3 Integrity checks and No-Deallocate After Sanitize

Another issue is availability of the data returned through the interface. Some of the sanitize operations (e.g., Block Erase) affect the physical devices in such a way that directly reading the accessible storage may trigger internal integrity checks resulting in error responses instead of returning the contents of the storage. Other sanitize operations (e.g., Crypto Erase) may scramble the vendor specific internal format of the data also resulting in error responses instead of returning the contents of the storage.

Some devices compensate for these issues by performing an additional internal write operation on all storage that is able to be allocated for user data. However, this has the side effect of potentially significant additional wear on the device as well as the side effect of obscuring the results of the initial sanitize operation (i.e., the writes forensically destroy the ability to audit the result of the initial sanitize operation). Given this side effect, process audits of sanitize behavior only prove effective results when the No-Deallocate After Sanitize bit is set the same way (e.g., set to '1') for both process audits and the individual device audits.

The Sanitize command introduced in NVM Express Base Specification revision 1.3 included a mechanism to specify that sanitized addressable storage not be deallocated, thereby allowing observations of the results of the sanitization operation. However, some architectures and products (e.g., integrity checking circuitry) interact with this capability in such a way as to defeat the sanitize result observability purpose. New features were added to NVM Express Base Specification revision 1.4 that include extended information about the sanitization capabilities of devices, a new asynchronous event, and configuration of the response to No-Deallocate After Sanitize requests. These features are intended to both support new systems that understand the new capabilities, as well to help manage legacy systems that do not understand the new capabilities without losing the ability to sanitize as requested.

A.4 Bad Media and Vendor Specific NAND Use

Another audit capability that is not supported by NVM Express is checking that any media that could not be sanitized (e.g., bad physical blocks) has been removed from the pool of storage that is able to be used as addressable storage.

An approach that is performed under some circumstances is removing the storage components from the NVM Express device after a sanitize operation and reading the contents in laboratory conditions. However, this approach also has multiple difficulties. When physical storage devices are removed from a NVM Express device, much context is lost. This includes:

a) any encoding for zero's/one's balance;
b) identification of which components contain device firmware or other non-data information; and

c) which media has been retired and cannot be sanitized.

Annex B. Host Considerations (Informative)

B.1 Basic Steps when Building a Command

When host software builds a command for the controller to execute, it first checks to make sure that the appropriate Submission Queue (SQ) is not full. The Submission Queue is full when the number of entries in the queue is one less than the queue size. Once an empty slot (pFreeSlot) is available:

1. Host software builds a command at SQ[pFreeSlot] with:
    a. CDW0.OPC is set to the appropriate command to be executed by the controller;
    b. CDW0.FUSE is set to the appropriate value, depending on whether the command is a fused operation;
    c. CDW0.CID is set to a unique identifier for the command when combined with the Submission Queue identifier;
    d. The Namespace Identifier, NSID field, is set to the namespace the command applies to;
    e. MPTR shall be filled in with the offset to the beginning of the Metadata Region, if there is a data transfer and the namespace format contains metadata as a separate buffer;
    f. PRP1 and/or PRP2 (or SGL Entry 1 if SGLs are used) are set to the source/destination of data transfer, if there is a data transfer; and
    g. CDW10 – CDW15 are set to any command specific information;

Host software then completes a transport specific action in order to submit the command for processing.

B.2 Creating an I/O Submission Queue

This example describes how host software creates an I/O Submission Queue that utilizes non-contiguous PRP entries. Creating an I/O Submission Queue that utilizes a PRP List is only valid if the controller supports non-contiguous queues as indicated in CAP.CQR.

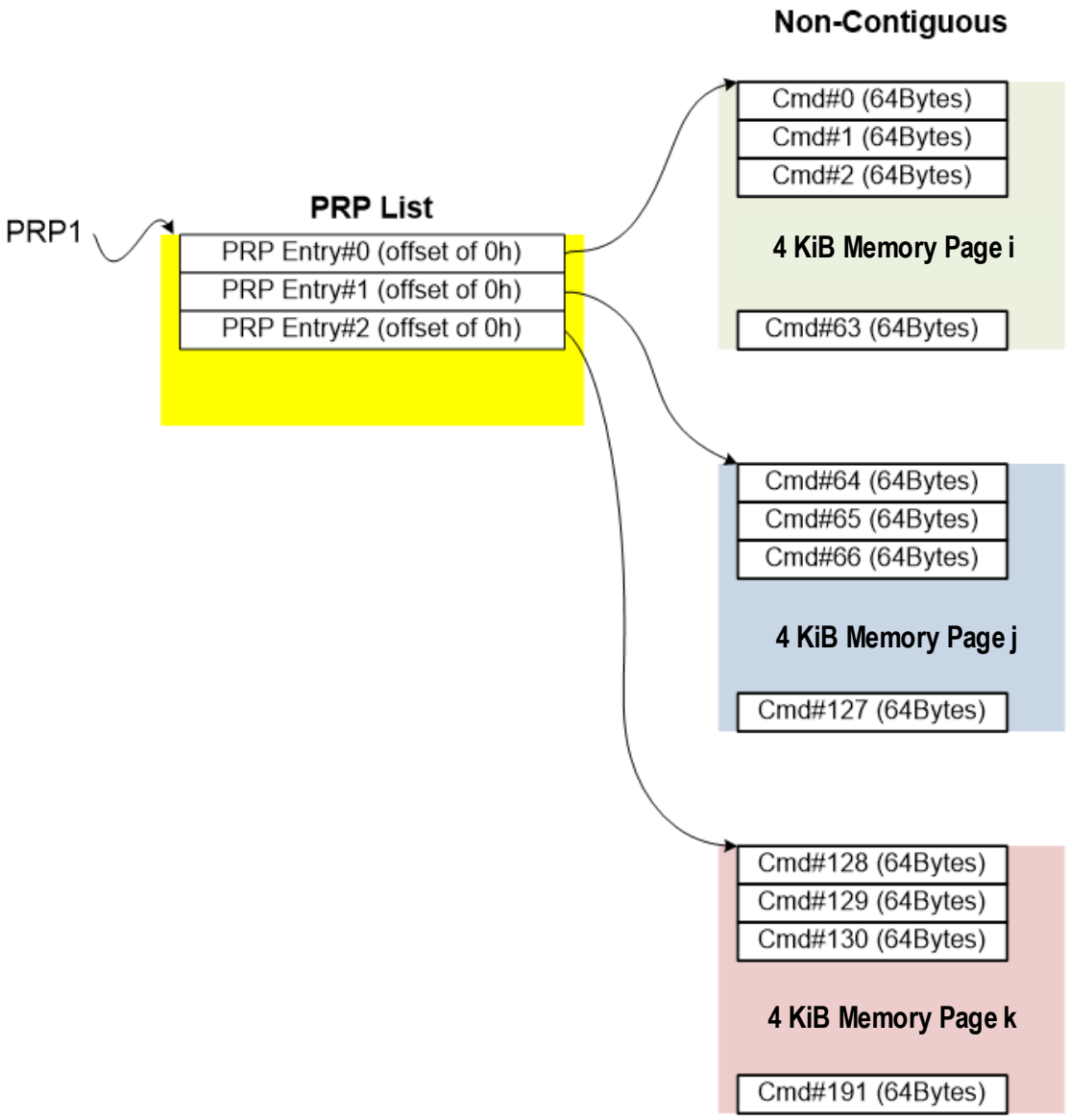
Prior to creating an I/O Submission Queue, host software shall create the I/O Completion Queue that the SQ uses with the Create I/O Completion Queue command.

To create an I/O Submission Queue, host software builds a Create I/O Submission Queue command for the Admin Submission Queue. Host software builds the Create I/O Submission Queue command in the next free Admin Submission Queue command location. The attributes of the command are:

- CDW0.OPC is set to 01h;
- CDW0.FUSE is cleared to 00b indicating that this is not a fused operation;
- CDW0.CID is set to a free command identifier;
- The NSID field is cleared to 0h; Submission Queues are not specific to a namespace;
- MPTR is cleared to 0h; metadata is not used for this command;
- PRP1 is set to the physical address of the PRP List. The PRP List is shown in Figure 479 for a PRP List with three entries;
- PRP2 is cleared to 0h; PRP Entry 2 is not used for this command;
- CDW10.QSIZE is set to the size of queue to create. In this case, the value is set to 191, indicating a queue size of 192 entries. The queue size shall not exceed the maximum queue entries supported, indicated in the CAP.MQES field;
- CDW10.QID is set to the Submission Queue identifier;
- CDW11.CQID is set to the I/O Completion Queue identifier where command completions are posted;
- CDW11.QPRIO is set to 10b, indicating a Medium priority queue; and
- CDW11.PC is cleared to ‘0’ indicating that the data buffer indicated by PRP1 is not physically contiguously.

Host software then completes a transport specific action in order to submit the command for processing. Host software shall maintain the PRP List unmodified in host memory until the Submission Queue is deleted.

Figure 479: PRP List Describing I/O Submission Queue



B.3 Executing a Fused Operation

This example describes how host software creates and executes a fused command, specifically Compare and Write for a total of 16 KiB of data. In this case, there are two commands that are created. The first command is the Compare, referred to as CMD0. The second command is the Write, referred to as CMD1. In this case, end-to-end data protection is not enabled and the size of each logical block is 4 KiB.

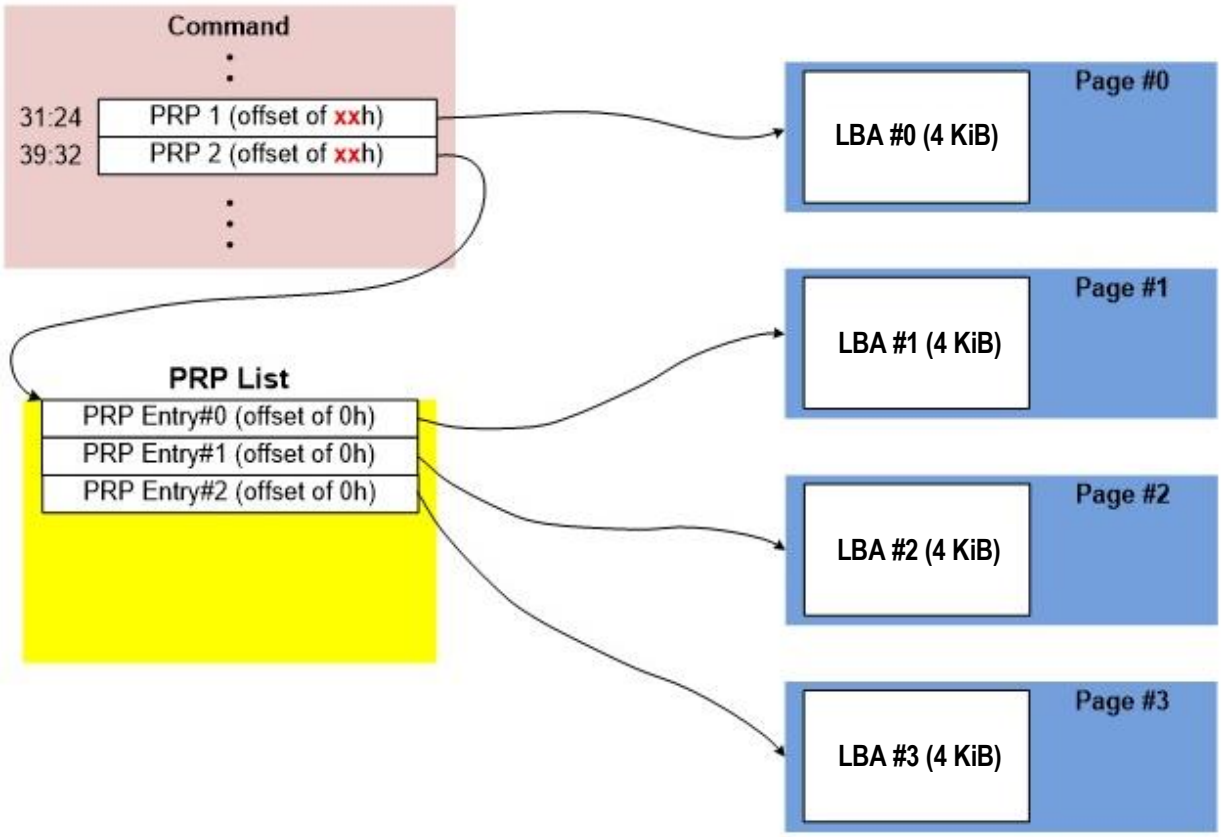
To build commands for a fused operation, host software utilizes two available adjacent command locations in the appropriate I/O Submission Queue as is described in section 3.4.2.

The attributes of the Compare command are:

- CMD0.CDW0.OPC is set to 05h for Compare;
- CMD0.CDW0.FUSE is set to 01b indicating that this is the first command of a fused operation;
- CMD0.CDW0.CID is set to a free command identifier;
- CMD0.NSID is set to identify the appropriate namespace;

- If metadata is being used in a separate buffer, then the location of that buffer is specified in the CMD0.MPTR field;
- The physical address of the first page of the data to compare:
    - If PRPs are used, CMD0.PRP1 is set to the physical address of the first page of the data to compare and CMD0.PRP2 is set to the physical address of the PRP List. The PRP List is shown in Figure 480 for a PRP List with three entries; or
    - If the command uses SGLs, CMD0.SGL1 is set to an appropriate SGL segment descriptor depending on whether more than one descriptor is needed;
- CMD0.CDW10.SLBA is set to the first LBA to compare against. Note that this field also spans Command Dword 11;
- CMD0.CDW12.LR is cleared to '0' to indicate that the controller should apply all available error recovery means to retrieve the data for comparison;
- CMD0.CDW12.FUA is cleared to '0', indicating that the data may be read from any location, including a volatile cache, in the NVM subsystem;
- CMD0.CDW12.PRINFO is cleared to 0h since end-to-end protection is not enabled;
- CMD0.CDW12.NLB is set to 3h, indicating that four logical blocks of a size of 4 KiB each are to be compared against;
- CMD0.CDW14 is cleared to 0h since end-to-end protection is not enabled; and
- CMD0.CDW15 is cleared to 0h since end-to-end protection is not enabled.

Figure 480: PRP List Describing Data to Compare



The attributes of the Write command are:

- CMD1.CDW0.OPC is set to 01h for Write;
- CMD1.CDW0.FUSE is set to 10b indicating that this is the second command of a fused operation;
- CMD1.CDW0.CID is set to a free command identifier;

- CMD1.NSID is set to identify the appropriate namespace. This value shall be the same as CMD0.NSID;
- If metadata is being used in a separate buffer, then the location of that buffer is specified in the CMD1.MPTR field;
- The physical address of the first page of data to write is identified:
    - If the command uses PRPs, then CMD1.PRP1 is set to the physical address of the first page of the data to write and CMD1.PRP2 is set to the physical address of the PRP List. The PRP List includes three entries; or
    - If the command uses SGLs, CMD1.SGL1 is set to an appropriate SGL segment descriptor depending on whether more than one descriptor is needed;
- CMD1.CDW10.SLBA is set to the first LBA to compare against. Note that this field also spans Command Dword 11. This value shall be the same as CMD0.CDW10.SLBA;
- CMD1.CDW12.LR is cleared to '0' to indicate that the controller should apply all available error recovery means to write the data to the NVM;
- CMD1.CDW12.FUA is cleared to '0', indicating that the data may be written to any location, including a volatile cache, in the NVM subsystem;
- CMD1.CDW12.PRINFO is cleared to 0h since end-to-end protection is not enabled;
- CMD1.CDW12.NLB is set to 3h, indicating that four logical blocks of a size of 4 KiB each are to be compared against. This value shall be the same as CMD0.CDW12.NLB;
- CMD1.CDW14 is cleared to 0h since end-to-end protection is not enabled; and
- CMD1.CDW15 is cleared to 0h since end-to-end protection is not enabled.

Host software then completes a transport specific action in order to submit the command for processing. Note that the transport specific submit action shall indicate both commands have been submitted at one time.

B.4 Asynchronous Event Request Host Software Recommendations

This section describes the recommended host software procedure for Asynchronous Event Requests.

The host sends *n* Asynchronous Event Request commands (refer to section 3.5.1, step 12). When an Asynchronous Event Request completes (providing Event Type, Event Information, and Log Page details):

- If the event(s) in the reported Log Page may be disabled with the Asynchronous Event Configuration feature (refer to section 5.27.1.8), then host software issues a Set Features command for the Asynchronous Event Configuration feature specifying to disable reporting of all events that utilize the Log Page reported. Host software should wait for the Set Features command to complete;
- Host software issues a Get Log Page command requesting the Log Page reported as part of the Asynchronous Event Command completion. Host software should wait for the Get Log Page command to complete;
- Host software parses the returned Log Page. If the condition is not persistent, then host software should re-enable all asynchronous events that utilize the Log Page. If the condition is persistent, then host software should re-enable all asynchronous events that utilize the Log Page except for the one(s) reported in the Log Page. The host re-enables events by issuing a Set Features command for the Asynchronous Event Configuration feature;
- Host software should issue an Asynchronous Event Request command to the controller (restoring to *n* the number of these commands outstanding); and
- If the reporting of event(s) was disabled, host software should enable reporting of the event(s) using the Asynchronous Event Configuration feature. If the condition reported may persist, host software should continue to monitor the event (e.g., spare below threshold) to determine if reporting of the event should be re-enabled.

B.5 Updating Controller Doorbell Properties using a Shadow Doorbell Buffer

B.5.1. Shadow Doorbell Buffer Overview

Controllers that support the Doorbell Buffer Config command are typically emulated controllers where this feature is used to enhance the performance of host software running in Virtual Machines. If supported by the controller, host software may enable Shadow Doorbell buffers by submitting the Doorbell Buffer Config command (refer to section 5.8).

After the completion of the Doorbell Buffer Config command, host software shall submit commands by updating the appropriate entry in the Shadow Doorbell buffer instead of updating the controller's corresponding doorbell property. If updating an entry in the Shadow Doorbell buffer changes the value from being less than or equal to the value of the corresponding EventIdx buffer entry to being greater than that value, then the host shall also update the controller's corresponding doorbell property to match the value of that entry in the Shadow Doorbell buffer. Queue wrap conditions shall be taken into account in all comparisons in this paragraph.

The controller may read from the Shadow Doorbell buffer and update the EventIdx buffer at any time (e.g., before the host writes to the controller's doorbell property).

B.5.2. Example Algorithm for Controller Doorbell Property Updates

Host software may use modular arithmetic where the modulus is the queue depth to decide if the controller doorbell property should be updated, specifically:

- Compute *X* as the new doorbell value minus the corresponding EventIdx value, modulo queue depth; and
- Compute *Y* as the new doorbell value minus the old doorbell value in the shadow doorbell buffer, also modulo queue depth.

If *X* is less than or equal to *Y*, the controller doorbell property should be updated with the new doorbell value.