NVM Express

NVM Express

Revision 1.3

May 1, 2017

Please send comments to info@nvmexpress.org

NVM Express revision 1.3 specification available for download at http://nvmexpress.org. NVM Express revision 1.3 ratified on April 26, 2017.

SPECIFICATION DISCLAIMER

LEGAL NOTICE:

This NVM Express revision 1.3 specification 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 revision 1.3 specification subject, however, to the Member's continued compliance with the Company's Intellectual Property Policy and Bylaws and the Member's Participation Agreement.

NOTICE TO NON-MEMBERS OF NVM EXPRESS, INC.: If you are not a Member of NVM Express, Inc. and you have obtained a copy of this document, you only have a right to review this document or make reference to or cite this document. Any such references or citations to this document must acknowledge NVM Express, Inc. copyright ownership of this document. The proper copyright citation or reference is as follows: **"© 2007 - 2017 NVM Express, Inc. ALL RIGHTS RESERVED."** When making any such citations or references to this document you are not permitted to revise, alter, modify, make any derivatives of, or otherwise amend the referenced portion of this document in any way without the prior express written permission of NVM Express, Inc. Nothing contained in this document shall be deemed as granting you any kind of license to implement or use this document or the specification described therein, or any of its contents, either expressly or impliedly, or to any intellectual property owned or controlled by NVM Express, Inc., including, without limitation, any trademarks of NVM Express, Inc.

LEGAL DISCLAIMER:

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

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

NVM Express Workgroup
c/o Virtual, Inc.
401 Edgewater Place, Suite 600
Wakefield, MA 01880
info@nvmexpress.org

Table of Contents

1 Introduction

1.1 Overview

NVM Express (NVMe) is an interface that allows host software to communicate with a non-volatile memory subsystem. This interface is optimized for Enterprise and Client solid state drives, typically attached as a register level interface to the PCI Express interface.

Note: During development, this specification was referred to as Enterprise NVMHCI. However, the name was modified to NVM Express prior to specification completion. This interface is targeted for use in both Client and Enterprise systems.

For an overview of changes from revision 1.2.1 to revision 1.3, refer to nvmexpress.org/changes for a document that describes the new features, including mandatory requirements for a controller to comply with revision 1.3.

1.1.1 NVMe over PCIe and NVMe over Fabrics

NVM Express 1.3 and prior revisions define a register level interface for host software to communicate with a non-volatile memory subsystem over PCI Express (NVMe over PCIe). The NVMe over Fabrics specification defines a protocol interface and related extensions to NVMe that enable operation over other interconnects (e.g., Ethernet, InfiniBand™, Fibre Channel). The NVMe over Fabrics specification has an NVMe Transport binding for each NVMe Transport (either within that specification or by reference).

In this specification a requirement/feature may be documented as specific to NVMe over Fabrics or to a particular NVMe Transport binding. In addition, support requirements for features and functionality may differ between NVMe over PCIe and NVMe over Fabrics.

To comply with NVM Express 1.2.1, a controller shall support the NVM Subsystem NVMe Qualified Name in the Identify Controller data structure in Figure 109.

1.2 Scope

The specification defines a register interface for communication with a non-volatile memory subsystem. It also defines a standard command set for use with the NVM subsystem.

1.3 Outside of Scope

The register 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 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 (for example, PCI, PCI Express and PCI-X).

1.4 Theory of Operation

NVM Express is a scalable host controller interface designed to address the needs of Enterprise and Client systems that utilize PCI Express based solid state drives. 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 64K 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 is necessary in the command submission path.
- Support for up to 65,535 I/O queues, with each I/O queue supporting up to 64K outstanding commands.
- Priority associated with each I/O queue with well-defined arbitration mechanism.
- All information to complete a 4KB 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.
- Support for multi-path I/O and namespace sharing.

This specification defines a streamlined set of registers whose functionality includes:

- 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)
- Doorbell registers for scalable number of Submission and Completion Queues

An NVM Express controller is associated with a single PCI Function. The capabilities and settings that apply to the entire controller are indicated in the Controller Capabilities (CAP) register and the Identify Controller data structure.

A namespace is a quantity of non-volatile memory that may be formatted into logical blocks. 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 and Namespace Attachment commands. The Identify Namespace data structure indicates capabilities and settings that are specific to a particular namespace. The capabilities and settings that are common to all namespaces are reported by the Identify Namespace data structure for namespace ID FFFFFFFFh.

NVM Express 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. Multiple Submission Queues may utilize the same Completion Queue. Submission and Completion Queues are allocated in memory.

An Admin Submission 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 may be submitted to the Admin Submission Queue.

An I/O Command Set is used with an I/O queue pair. This specification defines one I/O Command Set, named the NVM Command Set. The host selects one I/O Command Set that is used for all I/O queue pairs.

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 1 provides a graphical representation of the queue pair mechanism, showing a 1:1 mapping between Submission Queues and Completion Queues. Figure 2 shows an example where multiple I/O Submission Queues utilize the same

I/O Completion Queue on Core B. Figure 1 and Figure 2 show that there is always a 1:1 mapping between the Admin Submission Queue and Admin Completion Queue.

Figure 1: Queue Pair Example, 1:1 Mapping

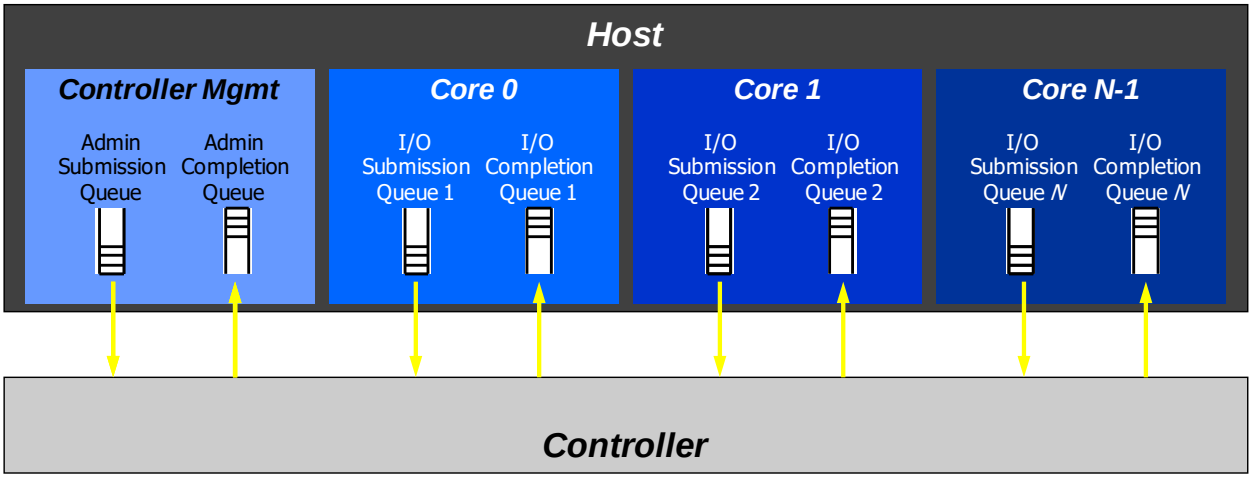
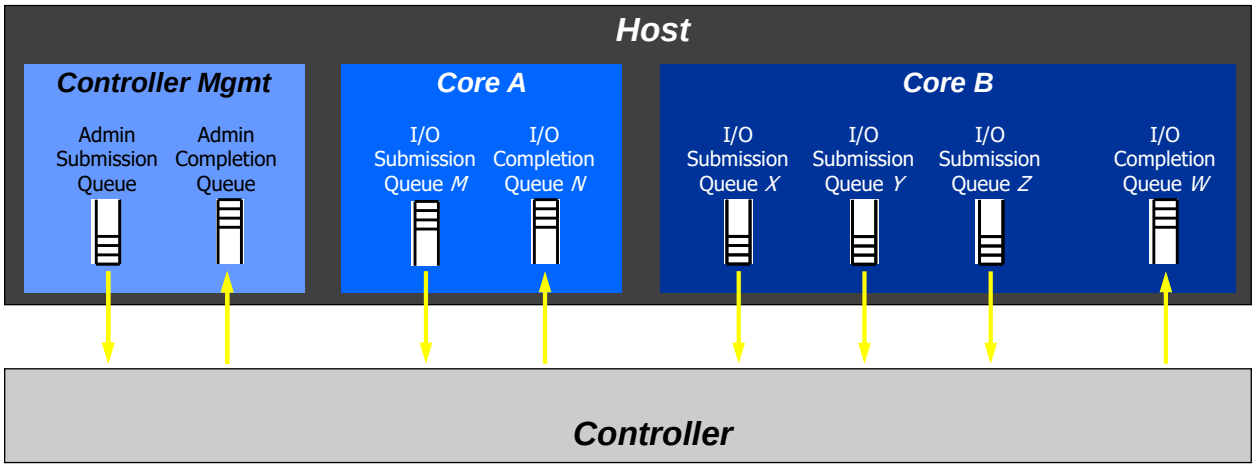


Figure 2: Queue Pair Example, *n*:1 Mapping



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, however, it may then 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. Each command may include two PRP entries or one Scatter Gather List (SGL) 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. Multiple Submission Queues may be associated with a single Completion Queue. This feature 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 it has processed 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 consulting a register. This enables host software to determine whether the new entry was posted as part of the previous or current round of completion notifications. Specifically, each round through the Completion Queue entries, the controller inverts the Phase Tag bit.

1.4.1 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 PCI Express 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. 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 3 shows an NVM subsystem that contains a single NVM Express controller 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 3 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 3: NVM Express Controller with Two Namespaces

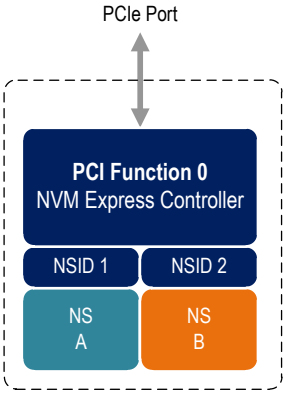
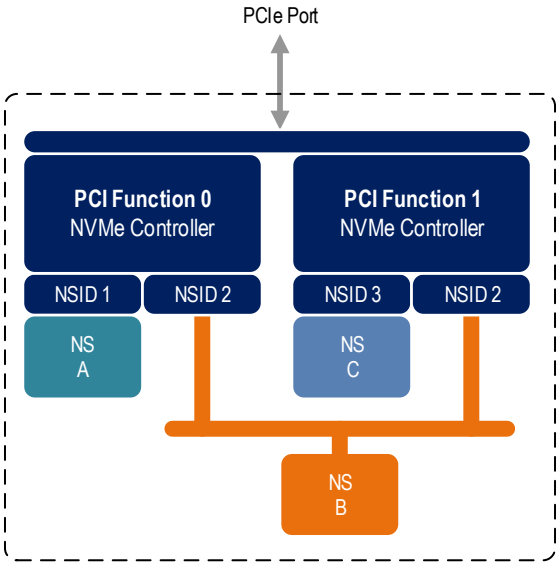


Figure 4 shows a multi-Function NVM Subsystem with a single PCI Express port containing two controllers, one controller is associated with PCI Function 0 and the other controller is associated with PCI Function 1. Each controller supports a single private namespace and access to shared namespace B. The 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 4: NVM Subsystem with Two Controllers and One Port

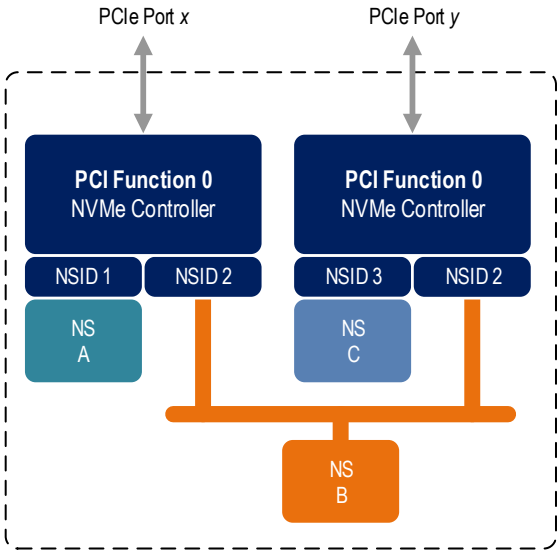


There is a unique Identify Controller data structure for each controller and a unique Identify Namespace data structure for each namespace. Controllers with access to a shared namespace return the Identify Namespace data structure associated with that shared namespace (i.e., the same data structure contents 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 7.10.

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 6.4). 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 5 illustrates an NVM Subsystem with two PCI Express ports, each with an associated controller. 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 4.

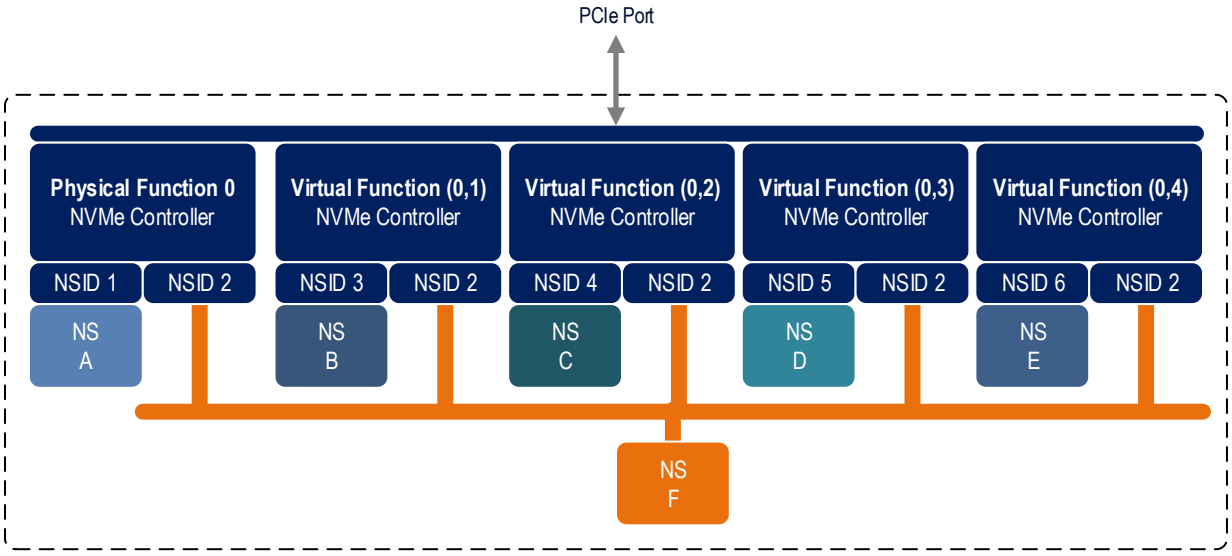
Figure 5: NVM Subsystem with Two Controllers and Two Ports



The two ports shown in Figure 5 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 6 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 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 8.5.4 for more information on SR-IOV.

Figure 6: 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.

1.5 Conventions

Hardware shall return '0' for all bits and registers that are marked as reserved, and host software shall write all reserved bits and registers with the value of '0'.

Inside the register section, the following abbreviations are used:

RO	Read Only
RW	Read Write
R/W	Read Write. The value read may not be the last value written.
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. The value is initialized at reset, for example by an expansion ROM, or in the case of integrated devices, by a platform BIOS.

For some register 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 register field is referred to in the document, the convention used is "Register Symbol.Field Symbol". For example, the PCI command register parity error response enable field is referred to by the name CMD.PEE. If the register field is an array of bits, the field is referred to as "Register Symbol.Field Symbol (array offset to element)".A 0-based value is a numbering scheme for which the number 0h actually corresponds to a value of 1h and thus produces the pattern of 0h = 1h, 1h = 2h, 2h = 3h, etc. In this numbering scheme, there is not a method for specifying the value of 0h. Values in this specification are 1-based (i.e., the number 1h corresponds to a value of 1h, 2h=2h, etc.) unless otherwise specified.

When a size is stated in the document as KB, the convention used is 1KB = 1024 bytes.

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.

1.6 Definitions

1.6.1 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.

1.6.2 arbitration burst

The maximum number of commands that may be launched at one time from a Submission Queue that is using round robin or weighted round robin with urgent priority class arbitration.

1.6.3 arbitration mechanism

The method used to determine which Submission Queue is selected next to launch commands for execution by the controller. Three arbitration mechanisms are defined including round robin, weighted round robin with urgent priority class, and vendor specific. Refer to section 4.11.

1.6.4 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.

1.6.5 candidate command

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

1.6.6 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.

1.6.7 command submission

For NVMe over PCIe, 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 NVMe over Fabrics, refer to section 1.4.14 in the NVMe over Fabrics 1.0 specification.

1.6.8 controller

A PCI Express function that implements NVM Express.

1.6.9 directive

A method of host and NVM subsystem or controller information exchange. 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 9.

1.6.10 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).

1.6.11 extended LBA

An extended LBA is a larger LBA that is created when metadata associated with the LBA is transferred contiguously with the LBA data. Refer to Figure 255.

1.6.12 firmware slot

A firmware slot is a location in the controller used to store a firmware image. The controller stores between one and seven firmware images. When downloading new firmware to the controller, host software has the option of specifying which image is replaced by indicating the firmware slot number.

1.6.13 I/O command

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

1.6.14 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. I/O Completion Queue identifiers are from 1 to 65535.

1.6.15 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, Write for the NVM command set). I/O Submission Queue identifiers are from 1 to 65535.

1.6.16 LBA range

A collection of contiguous logical blocks specified by a starting LBA and number of logical blocks.

1.6.17 logical block

The smallest addressable data unit for Read and Write commands.

1.6.18 logical block address (LBA)

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

1.6.19 metadata

Metadata is contextual information about 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.

1.6.20 namespace

A quantity of non-volatile memory that may be formatted into logical blocks. When formatted, a namespace of size n is a collection of logical blocks with logical block addresses from 0 to (n-1).

1.6.21 Namespace ID (NSID)

An identifier used by a controller to provide access to a namespace. Refer to section 6.1 for the definitions of valid NSID, invalid NSID, active NSID, inactive NSID, allocated NSID, and unallocated NSID.

1.6.22 NVM

NVM is an acronym for non-volatile memory.

1.6.23 NVM subsystem

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

1.6.24 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 8.5).

A PCI Express SR-IOV Physical Function that supports NVM Express and the Virtualization Enhancements capability is an example of a primary controller (refer to section 8.5.4).

1.6.25 private namespace

A namespace that may only be attached to one controller at a time. A host may determine whether a namespace is a private namespace or may be a shared namespace by the value of the Namespace Multi-path I/O and Namespace Sharing Capabilities (NMIC) field in the Identify Namespace data structure.

1.6.26 privileged actions

An action (command, register write, etc.) that affects or has the potential to affect the state of the entire NVM subsystem and not only the controller and/or namespace with which the action is associated. Admin commands that are privileged include Namespace Management, Namespace Attachment, Virtualization Management, Format NVM, and Sanitize. A privileged register action is NVM subsystem reset. Vendor specific commands and registers may also be privileged.

1.6.27 Runtime D3 (Power Removed)

In Runtime D3 (RTD3) main power is removed from the controller. Auxiliary power may or may not be provided.

1.6.28 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 not possible.

1.6.29 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 8.5).

A PCI Express SR-IOV Virtual Function that supports NVM Express and receives resources from a primary controller is an example of a secondary controller (refer to section 8.5.4).

1.6.30 shared namespace

A namespace that may be attached to two or more controllers in an NVM subsystem concurrently. A host may determine whether a namespace is a private namespace or may be a shared namespace by the value of the Namespace Multi-path I/O and Namespace Sharing Capabilities (NMIC) field in the Identify Namespace data structure.

1.6.31 user data

Data that is composed of logical block data, metadata, and protection information.

1.7 Keywords

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

1.7.1 mandatory

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

1.7.2 may

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

1.7.3 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.

1.7.4 R

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

1.7.5 reserved

A keyword referring to bits, bytes, words, fields, and opcode values that are set-aside for future standardization. Their use and interpretation may be specified by future extensions to this or other specifications. A reserved bit, byte, word, field, or register shall be cleared to zero, or in accordance with a future extension to this specification. The recipient 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 register field produces undefined results.

1.7.6 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.

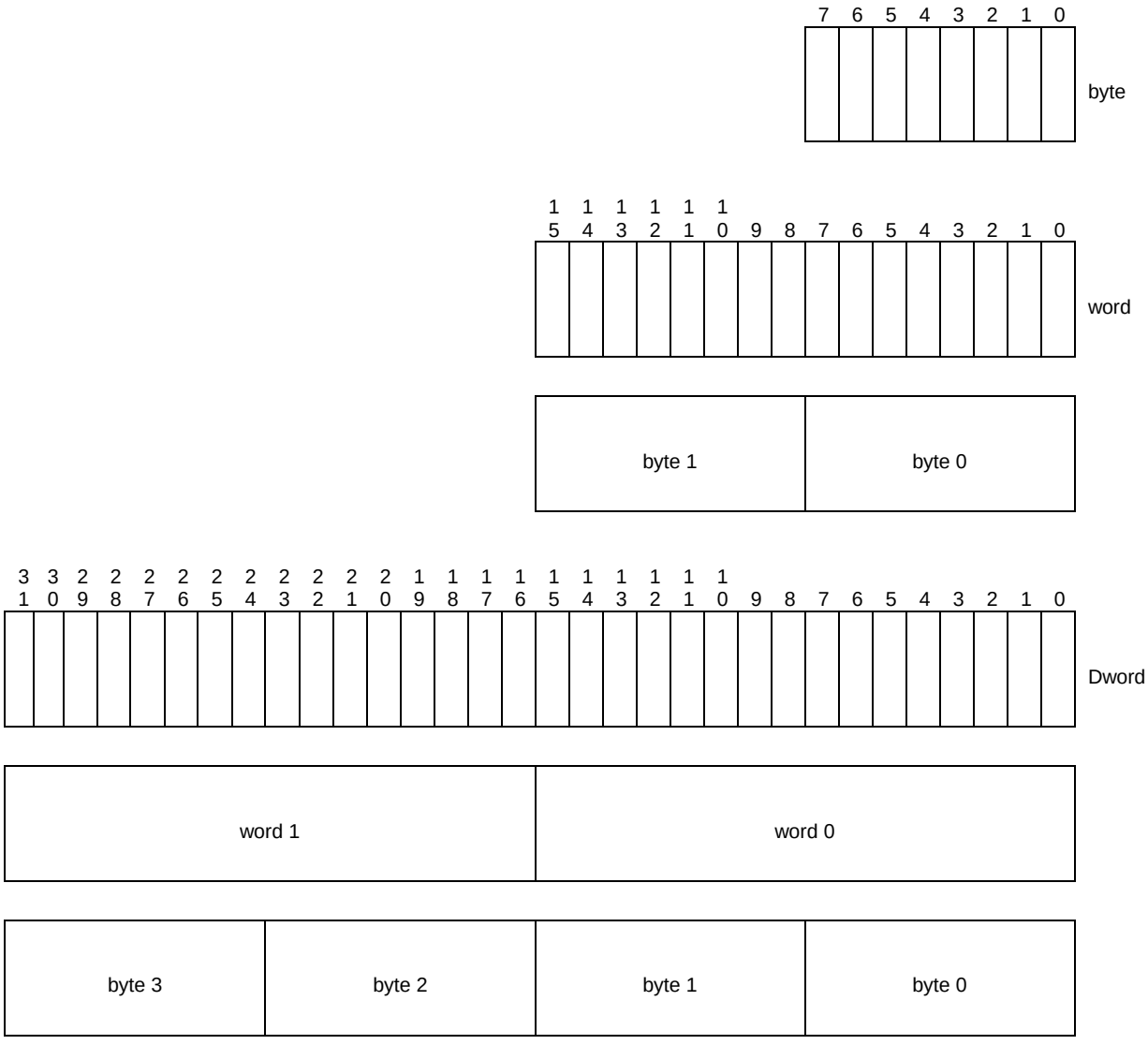
1.7.7 should

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

1.8 Byte, word and Dword Relationships

Figure 7 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. This specification specifies data in a little endian format.

Figure 7: Byte, word and Dword Relationships



1.9 References

INCITS 501-2016, Information technology – Security Features for SCSI Commands (SFSC). Available from http://webstore.ansi.org.

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

INCITS 522-2014, Information technology – ATA/ATAPI Command Set - 3 (ACS-3). Available from http://webstore.ansi.org.

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

NVM Express over Fabrics Specification, Revision 1.0. Available from http://www.nvmexpress.org.

NVM Express Management Interface Specification, Revision 1.0. Available from http://www.nvmexpress.org.

PCI specification, revision 3.0. Available from http://www.pcisig.com.

PCI Express specification, revision 2.1. Available from http://www.pcisig.com.

PCI Power Management specification. Available from http://www.pcisig.com.

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

PCI Firmware 3.0 specification. Available from http://www.pcisig.com.

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

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

UEFI 2.3.1 specification. Available from http://www.uefi.org.

Trusted Computing Group Storage Architecture Core specification, Version 2.01 Revision 1.00. Available from http://www.trustedcomputinggroup.org.

Trusted Computing Group Storage Interface Interactions Specification (SIIS). Available from http://www.trustedcomputinggroup.org.

1.10 References Under Development

ATA/ATAPI Command Set - 4 (ACS-4) [BSR INCITS 529]. Available from http://www.t13.org.

INCITS 506-201x, SCSI Block Commands - 4 (SBC-4)

2 System Bus (PCI Express) Registers

This section describes the PCI Express register values when the PCI Express is the system bus used. Other system buses may be used in an implementation. If a system bus is used that is not a derivative of PCI, then this section is not applicable.

This section details how the PCI Header, PCI Capabilities, and PCI Express Extended Capabilities should be constructed for an NVM Express controller. The fields shown are duplicated from the appropriate PCI or PCI Express specifications. The PCI documents are the normative specifications for these registers and this section details additional requirements for an NVM Express controller.

Start	End	Name	Type
00h	3Fh	PCI Header	
PMCAP	PMCAP+7h	PCI Power Management Capability	PCI Capability
MSICAP	MSICAP+9h	Message Signaled Interrupt Capability	PCI Capability
MSIXCAP	MSIXCAP+Bh	MSI-X Capability	PCI Capability
PXCAP	PXCAP+29h	PCI Express Capability	PCI Capability
AERCAP	AERCAP+47h	Advanced Error Reporting Capability	PCI Express Extended Capability

MSI-X is the recommended interrupt mechanism to use. However, some systems do not support MSI-X, thus devices should support both the MSI Capability and the MSI-X Capability.

It is recommended that implementations support the Advanced Error Reporting Capability to enable more robust error handling.

2.1 PCI Header

Start	End	Symbol	Name
00h	03h	ID	Identifiers
04h	05h	CMD	Command Register
06h	07h	STS	Device Status
08h	08h	RID	Revision ID
09h	0Bh	CC	Class Codes
0Ch	0Ch	CLS	Cache Line Size
0Dh	0Dh	MLT	Master Latency Timer
0Eh	0Eh	HTYPE	Header Type
0Fh	0Fh	BIST	Built In Self Test (Optional)
10h	13h	MLBAR (BAR0)	Memory Register Base Address, lower 32-bits <BAR0>
14h	17h	MUBAR (BAR1)	Memory Register Base Address, upper 32-bits <BAR1>
18h	1Bh	BAR2	Refer to section 2.1.12
1Ch	1Fh	BAR3	Vendor Specific
20h	23h	BAR4	Vendor Specific
24h	27h	BAR5	Vendor Specific
28h	2Bh	CCPTR	CardBus CIS Pointer
2Ch	2Fh	SS	Subsystem Identifiers
30h	33h	EROM	Expansion ROM Base Address (Optional)
34h	34h	CAP	Capabilities Pointer
35h	3Bh	R	Reserved
3Ch	3Dh	INTR	Interrupt Information
3Eh	3Eh	MGNT	Minimum Grant (Optional)
3Fh	3Fh	MLAT	Maximum Latency (Optional)

2.1.1 Offset 00h: ID - Identifiers

Bits	Type	Reset	Description
31:16	RO	Impl Spec	**Device ID (DID):** Indicates the device number assigned by the vendor. Specific to each implementation.
15:00	RO	Impl Spec	**Vendor ID (VID):** Indicates the company vendor, assigned by the PCI SIG.

2.1.2 Offset 04h: CMD - Command

Bit	Type	Reset	Description
15:11	RO	0	Reserved
10	RW	0	**Interrupt Disable (ID):** Disables the controller from generating pin-based INTx# interrupts. This bit does not have any effect on MSI operation.
09	RO	0	**Fast Back-to-Back Enable (FBE):** Not supported by NVM Express.
08	RW / RO	0	**SERR# Enable (SEE):** Controls error reporting.
07	RO	0	Hardwired to 0.
06	RW / RO	0	**Parity Error Response Enable (PEE):** When set to '1', the controller shall generate PERR# when a data parity error is detected. If parity is not supported, then this field is read-only '0'.
05	RO	0	**VGA Palette Snooping Enable (VGA):** Not supported by NVM Express.
04	RO	0	**Memory Write and Invalidate Enable (MWIE):** Not supported by NVM Express.
03	RO	0	**Special Cycle Enable (SCE):** Not supported by NVM Express.
02	RW	0	**Bus Master Enable (BME):** Enables the controller to act as a master for data transfers. When set to '1', bus master activity is allowed. When cleared to '0', the controller is not allowed to issue any Memory or I/O Requests.
01	RW	0	**Memory Space Enable (MSE)**: Controls access to the controller's register memory space.
00	RW	0	**I/O Space Enable (IOSE):** Controls access to the controller's target I/O space.

2.1.3 Offset 06h: STS - Device Status

Bit	Type	Reset	Description
15	RWC	0	**Detected Parity Error (DPE):** Set to '1' by hardware when the controller detects a parity error on its interface.
14	RWC/RO	0	**Signaled System Error (SSE):** Refer to the PCI SIG specifications.
13	RWC	0	**Received Master-Abort (RMA):** Set to '1' by hardware when the controller receives a master abort to a cycle it generated.
12	RWC	0	**Received Target Abort (RTA):** Set to '1' by hardware when the controller receives a target abort to a cycle it generated.
11	RO	0	**Signaled Target-Abort (STA):** Not supported by NVM Express.
10:09	RO	Impl Spec	**DEVSEL# Timing (DEVT):** Controls the device select time for the controller's PCI interface. This field is not applicable to PCI Express implementations.
08	RWC	0	**Master Data Pariy Error Detected (DPD):** Set to '1' by hardware when the controller, as a master, either detects a parity error or sees the parity error line asserted, and the parity error response bit (CMD.PEE) is set to '1'.
07	RO	Impl Spec	**Fast Back-to-Back Capable (FBC):** Indicates whether the controller accepts fast back-to-back cycles. This field is not applicable to PCI Express implementations.
06	RO	0	Reserved
05	RO	Impl Spec	**66 MHz Capable (C66):** Indicates whether the controller may operate at 66 MHz. This field is not applicable to PCI Express implementations.
04	RO	1	**Capabilities List (CL):** Indicates the presence of a capabilities list. The controller shall support the PCI Power Management capability as a minimum.
03	RO	0	**Interrupt Status (IS):** Indicates the interrupt status of the device ('1' = asserted).
02:00	RO	0	Reserved

2.1.4 Offset 08h: RID - Revision ID

Bits	Type	Reset	Description
07:00	RO	Impl Spec	**Revision ID (RID):** Indicates stepping of the controller hardware.

2.1.5 Offset 09h: CC - Class Code

Bits	Type	Reset	Description
23:16	RO	01h	**Base Class Code (BCC):** Indicates the base class code as a mass storage controller.
15:08	RO	08h	**Sub Class Code (SCC):** Indicates the sub class code as a Non-Volatile Memory controller.
07:00	RO	02h	**Programming Interface (PI):** This field specifies the programming interface of the controller is NVM Express. (Note: The PCI SIG documentation refers to this as Enterprise NVMHCI.)

2.1.6 Offset 0Ch: CLS – Cache Line Size

Bits	Type	Reset	Description
07:00	RW	00h	**Cache Line Size (CLS):** Cache Line Size register is set by the system firmware or operating system to the system cache size.

2.1.7 Offset 0Dh: MLT – Master Latency Timer

Bits	Type	Reset	Description
07:00	RO	00h	**Master Latency Timer (MLT):** Indicates the number of clocks the controller is allowed to act as a master on PCI. For a PCI Express device, this register does not apply and shall be hardwired to '0'.

2.1.8 Offset 0Eh: HTYPE – Header Type

Bits	Type	Reset	Description
07	RO	Impl Spec	**Multi-Function Device (MFD):** Indicates whether the controller is part of a multi-function device.
06:00	RO	00h	**Header Layout (HL):** Indicates that the controller uses a target device layout.

2.1.9 Offset 0Fh: BIST – Built In Self Test (Optional)

The following register is optional, but if implemented, shall look as follows. When not implemented, it shall be read-only 00h.

Bits	Type	Reset	Description
07	RO	Impl Spec	**BIST Capable (BC):** Indicates whether the controller has a BIST function.
06	RW	0	**Start BIST (SB):** Host software sets this bit to '1' to invoke BIST. The controller clears this bit to '0' when BIST is complete.
05:04	RO	00	Reserved
03:00	RO	0h	**Completion Code (CC):** Indicates the completion code status of BIST. A non-zero value indicates a failure.

2.1.10 Offset 10h: MLBAR (BAR0) – Memory Register Base Address, lower 32-bits

This register allocates space for the memory registers defined in section 3.

Bit	Type	Reset	Description
31:14	RW	0	**Base Address (BA):** Base address of register memory space. For controllers that support a larger number of doorbell registers or have vendor specific space following the doorbell registers, more bits are allowed to be RO such that more memory space is consumed.
13:04	RO	0	Reserved
03	RO	0	**Prefetchable (PF):** Indicates that this range is not pre-fetchable
02:01	RO	Impl Spec	**Type (TP):** Indicates where this range may be mapped. It is recommended to support mapping anywhere in 64-bit address space.
00	RO	0	**Resource Type Indicator (RTE):** Indicates a request for register memory space.

2.1.11 Offset 14h: MUBAR (BAR1) – Memory Register Base Address, upper 32-bits

This register specifies the upper 32-bit address of the memory registers defined in section 3.

Bit	Type	Reset	Description
31:00	RW	0	**Base Address (BA):** Upper 32-bits (bits 63:32) of the memory register base address.

NOTE: NVM Express implementations that reside behind PCI compliant bridges, such as PCI Express Endpoints, are restricted to having 32-bit assigned base address registers due to limitations on the maximum address that may be specified in the bridge for non-prefetchable memory. See the PCI Bridge 1.2 specification for more information on this restriction.

2.1.12 Offset 18h: BAR2 – Index/Data Pair Register Base Address or Vendor Specific (Optional)

If this register is configured as I/O space, then this register specifies the Index/Data Pair base address and is configured as shown in the table below. These registers are used to access the memory registers defined in section 3 using I/O based accesses.

Bit	Type	Reset	Description
31:03	RW	0	**Base Address (BA):** Base address of Index/Data Pair registers that is 8 bytes in size.
02:01	RO	0	Reserved
00	RO	1	**Resource Type Indicator (RTE):** Indicates a request for register I/O space.

If this register is configured as memory space (Resource Type Indicator is cleared to '0'), then the BAR2 register is vendor specific. Vendor specific space may also be allocated at the end of the memory registers defined in section 3.

2.1.13 Offset 1Ch – 20h: BAR3 – Vendor Specific

The BAR3 register is vendor specific. Vendor specific space may also be allocated at the end of the memory registers defined in section 3.

2.1.14 Offset 20h – 23h: BAR4 – Vendor Specific

The BAR4 register is vendor specific. Vendor specific space may also be allocated at the end of the memory registers defined in section 3.

2.1.15 Offset 24h – 27h: BAR5 – Vendor Specific

The BAR5 register is vendor specific. Vendor specific space may also be allocated at the end of the memory registers defined in section 3.

2.1.16 Offset 28h: CCPTR – CardBus CIS Pointer

Bit	Type	Reset	Description
31:00	RO	0	Not supported by NVM Express.

2.1.17 Offset 2Ch: SS - Sub System Identifiers

Bits	Type	Reset	Description
31:16	RO	HwInit	**Subsystem ID (SSID):** Indicates the sub-system identifier.
15:00	RO	HwInit	**Subsystem Vendor ID (SSVID):** Indicates the sub-system vendor identifier

2.1.18 Offset 30h: EROM – Expansion ROM (Optional)

If the register is not implemented, it shall be read-only 00h.

Bit	Type	Reset	Description
31:00	RW	Impl Spec	**ROM Base Address (RBA):** Indicates the base address of the controller's expansion ROM. Not supported for integrated implementations.

2.1.19 Offset 34h: CAP – Capabilities Pointer

Bit	Type	Reset	Description
7:0	RO	Impl Spec	**Capability Pointer (CP):** Indicates the first capability pointer offset.

2.1.20 Offset 3Ch: INTR - Interrupt Information

Bits	Type	Reset	Description
15:08	RO	Impl Spec	**Interrupt Pin (IPIN):** This indicates the interrupt pin the controller uses.
07:00	RW	00h	**Interrupt Line (ILINE):** Host software written value to indicate which interrupt line (vector) the interrupt is connected to. No hardware action is taken on this register.

2.1.21 Offset 3Eh: MGNT – Minimum Grant

Bits	Type	Reset	Description
07:00	RO	00h	**Grant (GNT):** Not supported by NVM Express.

2.1.22 Offset 3Fh: MLAT – Maximum Latency

Bits	Type	Reset	Description
07:00	RO	00h	**Latency (LAT):** Not supported by NVM Express.

2.2 PCI Power Management Capabilities

See section 3.1.5 for requirements when the PCI power management state changes.

Start	End	Symbol	Name
PMCAP	PMCAP+1h	PID	PCI Power Management Capability ID
PMCAP+2h	PMCAP+3h	PC	PCI Power Management Capabilities
PMCAP+4h	PMCAP+5h	PMCS	PCI Power Management Control and Status

2.2.1 Offset PMCAP: PID - PCI Power Management Capability ID

Bit	Type	Reset	Description
15:08	RO	Impl Spec	**Next Capability (NEXT):** Indicates the location of the next capability item in the list. This may be other capability pointers (such as Message Signaled Interrupts) or it may be the last item in the list.
07:00	RO	01h	**Cap ID (CID):** Indicates that this pointer is a PCI Power Management capability.

2.2.2 Offset PMCAP + 2h: PC – PCI Power Management Capabilities

Bit	Type	Reset	Description
15:11	RO	0h	**PME_Support (PSUP):** Not supported by NVM Express.
10	RO	0	**D2_Support (D2S):** Indicates support for the D2 power management state. Not recommended for implementation.
09	RO	0	**D1_Support (D1S):** Indicates support for the D1 power management state. Not recommended for implementation.
08:06	RO	000	**Aux_Current (AUXC):** Not supported by NVM Express.
05	RO	Impl Spec	**Device Specific Initialization (DSI):** Indicates whether device specific initialization is required.
04	RO	0	Reserved
03	RO	0	**PME Clock (PMEC):** Indicates that PCI clock is not required to generate PME#.
02:00	RO	Impl Spec	**Version (VS):** Indicates support for revision 1.2 or higher revisions of the PCI Power Management Specification.

2.2.3 Offset PMCAP + 4h: PMCS – PCI Power Management Control and Status

Bit	Type	Reset	Description
15	RWC	0	**PME Status (PMES):** Refer to the PCI SIG specifications.
14:13	RO	0	**Data Scale (DSC):** Refer to the PCI SIG specifications.
12:09	RO / RW	0	**Data Select (DSE):** If PME is not supported, then this field is read only '0'. Refer to the PCI SIG specifications.
08	RO / RW	0	**PME Enable (PMEE):** If PME is not supported, then this field is read only '0'. Refer to the PCI SIG specifications.
07:04	RO	0	Reserved
03	RO	1	**No Soft Reset (NSFRST):** A value of '1' indicates that the controller transitioning from D3hot to D0 because of a power state command does not perform an internal reset.
02	RO	0	Reserved
01:00	R/W	00	**Power State (PS):** This field is used both to determine the current power state of the controller and to set a new power state. The values are: 00 – D0 state 01 – D1 state 10 – D2 state 11 – $D3_{HOT}$ state When in the $D3_{HOT}$ state, the controller's configuration space is available, but the register I/O and memory spaces are not. Additionally, interrupts are blocked.

2.3 Message Signaled Interrupt Capability (Optional)

Start	End	Symbol	Name
MSICAP	MSICAP+1h	MID	Message Signaled Interrupt Capability ID
MSICAP+2h	MSICAP+3h	MC	Message Signaled Interrupt Message Control
MSICAP+4h	MSICAP+7h	MA	Message Signaled Interrupt Message Address
MSICAP+8h	MSICAP+Bh	MUA	Message Signaled Interrupt Upper Address
MSICAP+Ch	MSICAP+Dh	MD	Message Signaled Interrupt Message Data
MSICAP+10h	MSICAP+13h	MMASK	Message Signaled Interrupt Mask Bits (Optional)
MSICAP+14h	MSICAP+17h	MPEND	Message Signaled Interrupt Pending Bits (Optional)

2.3.1 Offset MSICAP: MID – Message Signaled Interrupt Identifiers

Bits	Type	Reset	Description
15:08	RO	Impl Spec	**Next Pointer (NEXT):** Indicates the next item in the list. This may be other capability pointers or it may be the last item in the list.
07:00	RO	05h	**Capability ID (CID):** Capabilities ID indicates this is a Message Signaled Interrupt (MSI) capability.

2.3.2 Offset MSICAP + 2h: MC – Message Signaled Interrupt Message Control

Bits	Type	Reset	Description
15:09	RO	0	Reserved
08	RO	Impl Spec	**Per-Vector Masking Capable (PVM):** Specifies whether controller supports MSI per-vector masking.
07	RO	1	**64 Bit Address Capable (C64):** Specifies whether the controller is capable of generating 64-bit messages. NVM Express controllers shall be 64-bit capable.
06:04	RW	000	**Multiple Message Enable (MME):** Indicates the number of messages the controller should assert. Controllers that only support single message MSI may implement this field as read-only.
03:01	RO	Impl Spec	**Multiple Message Capable (MMC):** Indicates the number of messages the controller wants to assert.
00	RW	0	**MSI Enable (MSIE):** If set to '1', MSI is enabled and the traditional interrupt pins are not used to generate interrupts. If cleared to '0', MSI operation is disabled and the traditional interrupt pins are used.

2.3.3 Offset MSICAP + 4h: MA – Message Signaled Interrupt Message Address

Bits	Type	Reset	Description
31:02	RW	0	**Address (ADDR):** Lower 32 bits of the system specified message address, always Dword aligned.
01:00	RO	00	Reserved

2.3.4 Offset MSICAP + 8h: MUA – Message Signaled Interrupt Upper Address

Bits	Type	Reset	Description
31:00	RW	0	**Upper Address (UADDR):** Upper 32 bits of the system specified message address. This register is required when the MSI Capability is supported by the controller.

2.3.5 Offset MSICAP + Ch: MD – Message Signaled Interrupt Message Data

Bits	Type	Reset	Description
15:00	RW	0	**Data (DATA):** This 16-bit field is programmed by system software if MSI is enabled. Its content is driven onto the lower word (PCI AD[15:0]) during the data phase of the MSI memory write transaction.

2.3.6 Offset MSICAP + 10h: MMASK – Message Signaled Interrupt Mask Bits (Optional)

Bits	Type	Reset	Description
31:00	RW	0	**Mask Bits (MASK):** For each Mask bit that is set to '1', the function is prohibited from sending the associated message.

2.3.7 Offset MSICAP + 14h: MPEND – Message Signaled Interrupt Pending Bits (Optional)

Bits	Type	Reset	Description
31:00	RW	0	**Pending Bits (PEND):** For each Pending bit that is set to '1', the function has a pending associated message.

2.4 MSI-X Capability (Optional)

Start	End	Symbol	Name
MSIXCAP	MSIXCAP+1h	MXID	MSI-X Capability ID
MSIXCAP+2h	MSIXCAP+3h	MXC	MSI-X Message Control
MSIXCAP+4h	MSIXCAP+7h	MTAB	MSI-X Table Offset and Table BIR
MSIXCAP+8h	MSIXCAP+Bh	MPBA	MSI-X PBA Offset and PBA BIR

Note: It is recommended that the controller allocate a unique MSI-X vector for each Completion Queue.

The Table BIR and PBA BIR data structures may be allocated in either BAR0-1 or BAR4-5 in implementations. These tables should be 4KB aligned. The memory page(s) that comprise the Table BIR and PBA BIR shall not include other registers/structures. It is recommended that these structures be allocated in BAR0-1 following the Submission Queue and Completion Queue Doorbell registers. Refer to the PCI reference for more information on allocation requirements for these data structures.

2.4.1 Offset MSIXCAP: MXID – MSI-X Identifiers

Bits	Type	Reset	Description
15:08	RO	Impl Spec	**Next Pointer (NEXT):** Indicates the next item in the list. This may be other capability pointers or it may be the last item in the list.
07:00	RO	11h	**Capability ID (CID):** Capabilities ID indicates this is an MSI-X capability.

2.4.2 Offset MSIXCAP + 2h: MXC – MSI-X Message Control

Bits	Type	Reset	Description
15	RW	0	**MSI-X Enable (MXE):** If set to '1' and the MSI Enable bit in the MSI Message Control register is cleared to '0', the function is permitted to use MSI-X to request service and is prohibited from using its INTx# pin (if implemented). If cleared to '0', the function is prohibited from using MSI-X to request service.
14	RW	0	**Function Mask (FM):** If set to '1', all of the vectors associated with the function are masked, regardless of their per vector Mask bit states. If cleared to '0', each vector's Mask bit determines whether the vector is masked or not. Setting or clearing the MSI-X Function Mask bit has no effect on the state of the per vector Mask bits.
13:11	RO	0h	Reserved
10:00	RO	Impl Spec	**Table Size (TS):** This value indicates the size of the MSI-X Table as the value *n*, which is encoded as *n* - 1. For example, a returned value of 3h corresponds to a table size of 4.

2.4.3 Offset MSIXCAP + 4h: MTAB – MSI-X Table Offset / Table BIR

Bits	Type	Reset	Description
31:03	RO	Impl Spec	**Table Offset (TO):** Used as an offset from the address contained by one of the function's Base Address registers to point to the base of the MSI-X Table. The lower three Table BIR bits are masked off (cleared to 000b) by system software to form a 32-bit Qword-aligned offset.
02:00	RO	Impl Spec	**Table BIR (TBIR):** This field indicates which one of a function's Base Address registers, located beginning at 10h in Configuration Space, is used to map the function's MSI-X Table into system memory. **BIR Value / BAR Offset** 0 – 10h 1 – na 2 – na 3 – Reserved 4 – 20h 5 – 24h 6 – Reserved 7 – Reserved For a 64-bit Base Address register, the Table BIR indicates the lower Dword. With PCI-to-PCI bridges, BIR values 2 through 5 are also reserved.

2.4.4 Offset MSIXCAP + 8h: MPBA – MSI-X PBA Offset / PBA BIR

Bits	Type	Reset	Description
31:03	RO	Impl Spec	**PBA Offset (PBAO):** Used as an offset from the address contained by one of the function's Base Address registers to point to the base of the MSI-X PBA. The lower three PBA BIR bits are masked off (cleared to 000b) by software to form a 32-bit Qword-aligned offset.
02:00	RO	Impl Spec	**PBA BIR (PBIR):** This field indicates which one of a function's Base Address registers, located beginning at 10h in Configuration Space, is used to map the function's MSI-X PBA into system memory. **BIR Value / BAR Offset** 0 – 10h 1 – na 2 – na 3 – Reserved 4 – 20h 5 – 24h 6 – Reserved 7 – Reserved

2.5 PCI Express Capability

The PCI Express Capability definitions below are based on the PCI Express 2.1 Base specification. Implementations may choose to base the device on a specification beyond the PCI Express 2.1 Base specification. In all cases, the PCI Express Base specification is the normative reference for the PCI Express Capability registers.

Note: TLP poisoning is a mandatory capability for PCI Express implementations. There are optional features of TLP poisoning, such as TLP poisoning for a transmitter. When an NVM Express controller has an error on a transmission to the host (e.g., error for a Read command), the error should be indicated as part of the NVM Express command status and not via TLP poisoning.

Start	End	Symbol	Name
PXCAP	PXCAP+1h	PXID	PCI Express Capability ID
PXCAP+2h	PXCAP+3h	PXCAP	PCI Express Capabilities
PXCAP+4h	PXCAP+7h	PXDCAP	PCI Express Device Capabilities
PXCAP+8h	PXCAP+9h	PXDC	PCI Express Device Control
PXCAP+Ah	PXCAP+Bh	PXDS	PCI Express Device Status
PXCAP+Ch	PXCAP+Fh	PXLCAP	PCI Express Link Capabilities
PXCAP+10h	PXCAP+11h	PXLC	PCI Express Link Control
PXCAP+12h	PXCAP+13h	PXLS	PCI Express Link Status
PXCAP+24h	PXCAP+27h	PXDCAP2	PCI Express Device Capabilities 2
PXCAP+28h	PXCAP+29h	PXDC2	PCI Express Device Control 2

2.5.1 Offset PXCAP: PXID – PCI Express Capability ID

Bits	Type	Reset	Description
15:8	RO	Impl Spec	**Next Pointer (NEXT):** Indicates the next item in the list. This may be other capability pointers or it may be the last item in the list.
7:0	RO	10h	**Capability ID (CID):** Indicates that this capability structure is a PCI Express capability.

2.5.2 Offset PXCAP + 2h: PXCAP – PCI Express Capabilities

Bits	Type	Reset	Description
15:14	RO	00b	Reserved
13:9	RO	Impl Spec	**Interrupt Message Number (IMN):** This field indicates the MSI/MSI-X vector that is used for the interrupt message generated in association with any of the status bits of this Capability structure. There are no status bits that generate interrupts defined in this capability within this specification, thus this field is not used.
8	RO	0h	**Slot Implemented (SI):** Not applicable for PCIe Express Endpoint devices.
7:4	RO	0h	**Device/Port Type (DPT):** Indicates the specific type of this PCI Express function. This device shall be indicated as a PCI Express Endpoint.
3:0	RO	2h	**Capability Version (VER):** Indicates that this capability structure is a PCI Express capability structure.

2.5.3 Offset PXCAP + 4h: PXDCAP – PCI Express Device Capabilities

Bits	Type	Reset	Description
31:29	RO	000b	Reserved
28	RO	1b	**Function Level Reset Capability (FLRC):** A value of '1' indicates the Function supports the optional Function Level Reset mechanism. NVM Express controllers shall support Function Level Reset.
27:26	RO	00b	**Captured Slot Power Limit Scale (CSPLS):** Specifies the scale used for the Slot Power Limit Value.
25:18	RO	0h	**Captured Slot Power Limit Value (CSPLV):** In combination with the Slot Power Limit Scale value, specifies the upper limit on power supplied by the slot. Power limit (in Watts) is calculated by multiplying the value in this field by the value in the Slot Power Limit Scale field.
17:16	RO	00b	Reserved
15	RO	1b	**Role-based Error Reporting (RER):** When set to '1', indicates that the Function implements role-based error reporting. This functionality is required.
14:12	RO	000b	Reserved

11:9	RO	Impl Spec	**Endpoint L1 Acceptable Latency (L1L):** This field indicates the acceptable latency that the Endpoint is able to withstand due to a transition from the L1 state to the L0 state.
08:06	RO	Impl Spec	**Endpoint L0s Acceptable Latency (L0SL):** This field indicates the acceptable total latency that the Endpoint is able to withstand due to the transition from L0s state to the L0 state.
05	RO	Impl Spec	**Extended Tag Field Supported (ETFS):** This field indicates the maximum supported size of the Tag field as a Requester.
04:03	RO	Impl Spec	**Phantom Functions Supported (PFS):** This field indicates the support for use of unclaimed Function Numbers to extend the number of outstanding transactions allowed by logically combining unclaimed Function Numbers with the Tag identifier.
02:00	RO	Impl Spec	**Max_Payload_Size Supported (MPS):** This field indicates the maximum payload size that the function may support for TLPs.

2.5.4 Offset PXCAP + 8h: PXDC – PCI Express Device Control

Bits	Type	Reset	Description
15	R/W	0b	**Initiate Function Level Reset –** A write of '1' initiates Function Level Reset to the Function. The value read by software from this bit shall always '0'.
14:12	RW/ RO	Impl Spec	**Max_Read_Request_Size (MRRS):** This field sets the maximum Read Request size for the Function as a Requester. The Function shall not generate Read Requests with size exceeding the set value.
11	RW/ RO	0	**Enable No Snoop (ENS):** If this field is set to '1', the Function is permitted to set the No Snoop bit in the Requestor Attributes of transactions it initiates that do not require hardware enforced cache coherency. This field may be hardwired to '0' if a Function would never set the No Snoop attribute in transactions it initiates.
10	RW/ RO	0	**AUX Power PM Enable (APPME):** If this field is set to '1', enables a Function to draw AUX power independent of PME AUX power. Functions that do not implement this capability hardware this bit to 0b.
09	RW/ RO	0	**Phantom Functions Enable (PFE):** If this field is set to '1', enables a Function to use unclaimed Functions as Phantom Functions to extend the number of outstanding transaction identifiers. If this field is cleared to '0', the Function is not allowed to use Phantom Functions.
08	RW/ RO	0	**Extended Tag Enable (ETE):** If this field is set to '1', enables a Function to use an 8-bit Tag field as a Requester. If this field is cleared to '0', the Function is restricted to a 5-bit Tag field.
07:05	RW/ RO	000b	**Max_Payload_Size (MPS):** This field sets the maximum TLP payload size for the Function. As a receiver, the Function shall handle TLPs as large as the set value. As a transmitter, the Function shall not generate TLPs exceeding the set value. Functions that support only the 128 byte max payload size are permitted to hardwire this field to 0h.
04	RW/ RO	Impl Spec	**Enable Relaxed Ordering (ERO):** If this field is set to '1', the Function is permitted to set the Relaxed Ordering bit in the Attributes field of transactions it initiates that do not require strong write ordering.
03	RW	0	**Unsupported Request Reporting Enable (URRE):** This bit, in conjunction with other bits, controls the signaling of Unsupported Requests by sending error messages.
02	RW	0	**Fatal Error Reporting Enable (FERE):** This bit, in conjunction with other bits, controls the signaling of Unsupported Requests by sending ERR_FATAL messages.
01	RW	0	**Non-Fatal Error Reporting Enable (NFERE):** This bit, in conjunction with other bits, controls the signaling of Unsupported Requests by sending ERR_NONFATAL messages.
00	RW	0	**Correctable Error Reporting Enable (CERE):** This bit, in conjunction with other bits, controls the signaling of Unsupported Requests by sending ERR_COR messages.

2.5.5 Offset PXCAP + Ah: PXDS – PCI Express Device Status

Bits	Type	Reset	Description
15:06	RO	0h	Reserved
05	RO	0	**Transactions Pending (TP):** When set to '1' this bit indicates that the Function has issued non-posted requests that have not been completed. This bit is cleared to '0' only when all outstanding non-posted requests have completed or have been terminated by the completion timeout mechanism. This bit shall also be cleared to '0' upon completion of a Function Level Reset.
04	RO	Impl Spec	**AUX Power Detected (APD):** Functions that require AUX power report this field as set to '1' if AUX power is detected by the Function.
03	RWC	0	**Unsupported Request Detected (URD):** When set to '1' this bit indicates that the function received an Unsupported Request. Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.
02	RWC	0	**Fatal Error Detected (FED):** When set to '1' this bit indicates that the status of fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.
01	RWC	0	**Non-Fatal Error Detected (NFED):** When set to '1' this bit indicates that the status of non-fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.
00	RWC	0	**Correctable Error Detected (CED):** When set to '1' this bit indicates status of correctable errors detected. Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.

2.5.6 Offset PXCAP + Ch: PXLCAP – PCI Express Link Capabilities

Bits	Type	Reset	Description
31:24	RO	HwInit	**Port Number (PN):** This field specifies the PCI Express port number for this device.
23	RO	0h	Reserved
22	RO	HwInit	**ASPM Optionality Compliance (AOC):** This field specifies Active State Power Management (ASPM) support
21	RO	0	**Link Bandwidth Notification Capability (LBNC):** Not applicable to Endpoints.
20	RO	0	**Data Link Layer Link Active Reporting Capable (DLLLA):** Not applicable to Endpoints.
19	RO	0	**Surprise Down Error Reporting Capable (SDERC):** Not applicable to Endpoints.
18	RO	Impl Spec	**Clock Power Management (CPM):** If this field is set to '1', the component tolerates the removal of any reference clock(s) via the "clock request" (CLKREQ#) mechanism when the Link is in the L1 and L2/L3 Ready Link states. If this field is cleared to '0', the component does not have this capability and that reference clock(s) shall not be removed in these Link states.
17:15	RO	Impl Spec	**L1 Exit Latency (L1EL):** This field indicates the L1 exit latency for the given PCI Express Link. The value reported indicates the length of time this port requires to complete transition from L1 to L0.
14:12	RO	Impl Spec	**L0s Exit Latency (L0SEL):** This field indicates the L0s exit latency for the given PCI Express Link. The value reported indicates the length of time this port requires to complete transition from L0s to L0.
11:10	RO	Impl Spec	**Active State Power Management Support (ASPMS):** This field indicates the level of ASPM supported on the given PCI Express Link.
09:04	RO	Impl Spec	**Maximum Link Width (MLW):** This field indicates the maximum Link width (x*n* – corresponding to *n* lanes) implemented by the component.
03:00	RO	Impl Spec	**Supported Link Speeds (SLS):** This field indicates the supported Link speed(s) of the associated port.

2.5.7 Offset PXCAP + 10h: PXLC – PCI Express Link Control

Bits	Type	Reset	Description
15:10	RO	0h	Reserved
09	RW/ RO	0h	**Hardware Autonomous Width Disable (HAWD):** When set to '1', disables hardware from changing the Link width for reasons other than attempting to correct unreliable Link operation by reducing Link width. Components that do not implement the ability autonomously to change Link width are permitted to hardwire this bit to '0'.
08	RW	0	**Enable Clock Power Management (ECPM):** When cleared to '0', clock power management is disabled and the device shall hold the CLKREQ# signal low. When set to '1', the device is permitted to use the CLKREQ# signal to power manage the Link clock according to the protocol defined for mini PCI Express.
07	RW	0	**Extended Synch (ES):** When set to '1', this bit forces the transmission of additional Ordered Sets when exiting the L0s state and when in the Recovery state. This mode provides external devices (e.g. logic analyzers) monitoring the Link time to achieve bit and symbol lock before the Link enters the L0 state and resumes communication.
06	RW	0	**Common Clock Configuration (CCC):** When set to '1', this bit indicates that this component and the component at the opposite end of this Link are operating with a distributed common reference clock. When cleared to '0' this component and the component at the opposite end of this Link are operating with asynchronous reference clocks.
05:04	RO	0h	Reserved: These bits are reserved on Endpoints.
03	RW	0	**Read Completion Boundary (RCB):** Indicate the RCB value of the root port.
02	RO	0	Reserved
01:00	RW	0h	**Active State Power Management Control (ASPMC):** This field controls the level of ASPM executed on the PCI Express Link.

2.5.8 Offset PXCAP + 12h: PXLS – PCI Express Link Status

Bits	Type	Reset	Description
15:13	RO	0h	Reserved
12	RO	Impl Spec	**Slot Clock Configuration:** If this bit is set to '1', it indicates that the component uses the same physical reference clock that the platform provides on the connector. If the device uses an independent clock irrespective of a reference on the connector, this bit shall be cleared to '0'.
11:10	RO	0h	Reserved
09:04	RO	na	**Negotiated Link Width (NLW):** This field indicates the negotiated Link width. This field is undefined when the Link is not up.
03:00	RO	na	**Current Link Speed (CLS):** This field indicates the negotiated Link speed. This field is undefined when the Link is not up.

2.5.9 Offset PXCAP + 24h: PXDCAP2 – PCI Express Device Capabilities 2

Bits	Type	Reset	Description
31:24	RO	0h	Reserved
23:22	RO	Impl Spec	**Max End-End TLP Prefixes (MEETP):** Indicates the maximum number of End-End TLP Prefixes supported by this Function. TLPs received by this Function that contain more End-End TLP Prefixes than are supported shall be handled as Malformed TLPs.
21	RO	Impl Spec	**End-End TLP Prefix Supported (EETPS):** Indicates whether End-End TLP Prefix support is offered by a Function. If cleared to '0', there is no support. If set to '1', the Function supports receiving TLPs containing End-End TLP Prefixes.
20	RO	Impl Spec	**Extended Fmt Field Supported (EFFS):** If set to '1', the Function supports the 3-bit definition of the Fmt field. If cleared to '0', the Function supports a 2-bit definition of the Fmt field.
19:18	RO	Impl Spec	**OBFF Supported (OBFFS):** This field indicates the level of support for OBFF.
17:14	RO	0h	Reserved
13:12	RO	Impl Spec	**TPH Completer Supported (TPHCS):** Defined encodings are listed in the following table. **Value** / **Definition** 00b – TPH and Extended TPH Completer not supported 01b – TPH Completer supported; Extended TPH Completer not supported 10b – Reserved 11b – Both TPH and Extended TPH Completer supported
11	RO	Impl Spec	**Latency Tolerance Reporting Supported (LTRS):** If set to '1', then the latency tolerance reporting mechanism is supported.
10	RO	0	**No RO-enabled PR-PR Passing (NPRPR):** Not applicable to NVM Express.
09	RO	Impl Spec	**128-bit CAS Completer Supported (128CCS):** This bit shall be set to '1' if the Function supports this optional capability.
08	RO	Impl Spec	**64-bit AtomicOp Completer Supported (64AOCS):** Includes FetchAdd, Swap, and CAS AtomicOps. This bit shall be set to '1' if the Function supports this optional capability.
07	RO	Impl Spec	**32-bit AtomicOp Completer Supported (32AOCS):** Includes FetchAdd, Swap, and CAS AtomicOps. This bit shall be set to '1' if the Function supports this optional capability.
06	RO	0	**AtomicOp Routing Supported (AORS):** Not applicable to NVM Express.
05	RO	0	**ARI Forwarding Supported (ARIFS):** Not applicable for NVM Express.
04	RO	1	**Completion Timeout Disable Supported (CTDS):** A value of '1' indicates support for the Completion Timeout Disable mechanism. The Completion Timeout Disable mechanism is required for Endpoints that issue requests on their own behalf.
03:00	RO	Impl Spec	**Completion Timeout Ranges Supported (CTRS):** This field indicates device function support for the optional Completion Timeout programmability mechanism.

2.5.10 Offset PXCAP + 28h: PXDC2 – PCI Express Device Control 2

Bits	Type	Reset	Description
31:15	RO	0h	Reserved
14:13	RW	Impl Spec	**OBFF Enable (OBFFE):** This field controls the capabilities enabled for OBFF.
12:11	RO	00b	Reserved
10	RW	0	**Latency Tolerance Reporting Mechanism Enable (LTRME):** When set to '1', enables the LTR mechanism. When cleared to '0', the LTR mechanism is disabled.
09:05	RO	0h	Reserved
04	RW	0	**Completion Timeout Disable (CTD):** When set to '1', this bit disables the Completion Timeout mechanism.
03:00	RW/RO	Impl Spec	**Completion Timeout Value:** Specifies the completion timeout value. If this feature is not supported in PXDCAP2, then this field is read only 0h.

2.6 Advanced Error Reporting Capability (Optional)

The Advanced Error Reporting definitions below are based on the PCI Express 2.1 Base specification. Implementations may choose to base the device on a specification beyond the PCI Express 2.1 Base specification. In all cases, the PCI Express Base specification is the normative reference for the Advanced Error Reporting registers.

Start	End	Symbol	Name
AERCAP	AERCAP+3h	AERID	AER Capability ID
AERCAP+4h	AERCAP+7h	AERUCES	AER Uncorrectable Error Status Register
AERCAP+8h	AERCAP+Bh	AERUCEM	AER Uncorrectable Error Mask Register
AERCAP+Ch	AERCAP+Fh	AERUCESEV	AER Uncorrectable Error Severity Register
AERCAP+10h	AERCAP+13h	AERCES	AER Correctable Error Status Register
AERCAP+14h	AERCAP+17h	AERCEM	AER Correctable Error Mask Register
AERCAP+18h	AERCAP+1Bh	AERCC	AER Advanced Error Capabilities and Control Reg.
AERCAP+1Ch	AERCAP+2Bh	AERHL	AER Header Log Register
AERCAP+38h	AERCAP+47h	AERTLP	AER TLP Prefix Log Register (Optional)

2.6.1 Offset AERCAP: AERID – AER Capability ID

Bits	Type	Reset	Description
31:20	RO	Impl Spec	**Next Pointer (NEXT):** Indicates the next item in the list. This may be other capability pointers or it may be the last item in the list.
19:16	RO	Impl Spec	**Capability Version (CVER):** Indicates the version of the capability structure. Reset value may be 1h or 2h.
15:0	RO	0001h	**Capability ID (CID):** Indicates that this capability structure is an Advanced Error Reporting capability.

2.6.2 Offset AERCAP + 4: AERUCES – AER Uncorrectable Error Status Register

This register indicates the error detection status of the individual errors on the controller. These bits are sticky – they are neither initialized nor modified during a hot reset or Function Level Reset (FLR).

Bits	Type	Reset	Description
31:26	RO	0	Reserved
25	RWC/RO	0	**TLP Prefix Blocked Error Status (TPBES)** (Optional)
24	RWC/RO	0	**AtomicOp Egress Blocked Status (AOEBS)** (Optional)
23	RWC/RO	0	**MC Blocked TLP Status (MCBTS)** (Optional)
22	RWC/RO	0	**Uncorrectable Internal Error Status (UIES)** (Optional)
21	RWC/RO	0	**ACS Violation Status (ACSVS)** (Optional)
20	RWC	0	**Unsupported Request Error Status (URES)**
19	RWC/RO	0	**ECRC Error Status (ECRCES)** (Optional)
18	RWC	0	**Malformed TLP Status (MTS)**
17	RWC/RO	0	**Receiver Overflow Status (ROS)** (Optional)
16	RWC	0	**Unexpected Completion Status (UCS)**
15	RWC/RO	0	**Completer Abort Status (CAS)** (Optional)
14	RWC	0	**Completion Timeout Status (CTS)**
13	RWC/RO	0	**Flow Control Protocol Error Status (FCPES)** (Optional)
12	RWC	0	**Poisoned TLP Status (PTS)**
11:05	RO	0	Reserved
04	RWC	0	**Data Link Protocol Error Status (DLPES)**
03:00	RO	0	Reserved

2.6.3 Offset AERCAP + 8: AERUCEM – AER Uncorrectable Error Mask Register

This register controls the reporting of the individual errors by the controller. A masked error is not reported in the Header Log register (AERHL), does not updated the First Error Pointer (AERCC.FEP), and is not reported to the host. These bits are sticky – they are neither initialized nor modified during a hot reset or FLR.

Bits	Type	Reset	Description
31:26	RO	0	Reserved
25	RW/RO	0	**TLP Prefix Blocked Error Mask (TPBEM)** (Optional)
24	RW/RO	0	**AtomicOp Egress Blocked Mask (AOEBM)** (Optional)
23	RW/RO	0	**MC Blocked TLP Mask (MCBTM)** (Optional)
22	RW/RO	1	**Uncorrectable Internal Error Mask (UIEM)** (Optional)
21	RW/RO	0	**ACS Violation Mask (ACSVM)** (Optional)
20	RW	0	**Unsupported Request Error Mask (UREM)**
19	RW/RO	0	**ECRC Error Mask (ECRCEM)** (Optional)
18	RW	0	**Malformed TLP Mask (MTM)**
17	RW/RO	0	**Receiver Overflow Mask (ROM)** (Optional)
16	RW	0	**Unexpected Completion Mask (UCM)**
15	RW/RO	0	**Completer Abort Mask (CAM)** (Optional)
14	RW	0	**Completion Timeout Mask (CTM)**
13	RW/RO	0	**Flow Control Protocol Error Mask (FCPEM)** (Optional)
12	RW	0	**Poisoned TLP Mask (PTM)**
11:05	RO	0	Reserved
04	RW	0	**Data Link Protocol Error Mask (DLPEM)**
03:00	RO	0	Reserved

2.6.4 Offset AERCAP + Ch: AERUCESEV – AER Uncorrectable Error Severity Register

This register controls whether an individual error is reported as a non-fatal or a fatal error. An error is reported as fatal when the corresponding error bit in the severity register is set (‘1’). If the bit is cleared (‘0’), the corresponding error is considered non-fatal. These bits are sticky – they are neither initialized nor modified during a hot reset or FLR.

Bits	Type	Reset	Description
31:26	RO	0	Reserved
25	RW/RO	0	**TLP Prefix Blocked Error Severity (TPBESEV)** (Optional)
24	RW/RO	0	**AtomicOp Egress Blocked Severity (AOEBSEV)** (Optional)
23	RW/RO	0	**MC Blocked TLP Severity (MCBTSEV)** (Optional)
22	RW/RO	1	**Uncorrectable Internal Error Severity (UIESEV)** (Optional)
21	RW/RO	0	**ACS Violation Severity (ACSVSEV)** (Optional)
20	RW	0	**Unsupported Request Error Severity (URESEV)**
19	RW/RO	0	**ECRC Error Severity (ECRCESEV)** (Optional)
18	RW	1	**Malformed TLP Severity (MTSEV)**
17	RW/RO	1	**Receiver Overflow Severity (ROSEV)** (Optional)
16	RW	0	**Unexpected Completion Severity (UCSEV)**
15	RW/RO	0	**Completer Abort Severity (CASEV)** (Optional)
14	RW	0	**Completion Timeout Severity (CTSEV)**
13	RW/RO	1	**Flow Control Protocol Error Severity (FCPESEV)** (Optional)
12	RW	0	**Poisoned TLP Severity (PTSEV)**
11:05	RO	0	Reserved
04	RW	1	**Data Link Protocol Error Severity (DLPESEV)**
03:00	RO	0	Reserved

2.6.5 Offset AERCAP + 10h: AERCS – AER Correctable Error Status Register

This register reports error status of individual correctable error sources from the controller. These bits are sticky – they are neither initialized nor modified during a hot reset or FLR.

Bits	Type	Reset	Description
31:16	RO	0	Reserved
15	RWC/RO	0	**Header Log Overflow Status (HLOS)** (Optional)
14	RWC/RO	0	**Corrected Internal Error Status (CIES)** (Optional)
13	RWC	0	**Advisory Non-Fatal Error Status (ANFES)**
12	RWC	0	**Replay Timer Timeout Status (RTS)**
11:09	RO	0	Reserved
08	RWC	0	**REPLAY_NUM Rollover Status (RRS)**
07	RWC	0	**Bad DLLP Status (BDS)**
06	RWC	0	**Bad TLP Status (BTS)**
05:01	RO	0	Reserved
00	RWC	0	**Receiver Error Status (RES)**

2.6.6 Offset AERCAP + 14h: AERCEM – AER Correctable Error Mask Register

This register controls the reporting of the individual correctable errors by the controller. A masked error is not reported to the host. These bits are sticky – they are neither initialized nor modified during a hot reset or FLR.

Bits	Type	Reset	Description
31:16	RO	0	Reserved
15	RW/RO	0	**Header Log Overflow Mask (HLOM)** (Optional)
14	RW/RO	0	**Corrected Internal Error Mask (CIEM)** (Optional)
13	RW	0	**Advisory Non-Fatal Error Mask (ANFEM)**
12	RW	0	**Replay Timer Timeout Mask (RTM)**
11:09	RO	0	Reserved
08	RW	0	**REPLAY_NUM Rollover Mask (RRM)**
07	RW	0	**Bad DLLP Mask (BDM)**
06	RW	0	**Bad TLP Mask (BTM)**
05:01	RO	0	Reserved
00	RW	0	**Receiver Error Mask (REM)**

2.6.7 Offset AERCAP + 18h: AERCC – AER Capabilities and Control Register

Bits	Type	Reset	Description
31:12	RO	0	Reserved
11	RO	0	**TLP Prefix Log Present (TPLP) :** If set to '1' and FEP is valid, this indicates that the TLP Prefix Log register contains valid information. This field is sticky – it is neither initialized nor modified during a hot reset or FLR.
10	RW/RO	0	**Multiple Header Recording Enable (MHRE):** If this field is set to '1', this enables the controller to generate more than one error header. This field is sticky – it is neither initialized nor modified during a hot reset or FLR. If the controller does not implement the associated mechanism, then this field is cleared to '0'.
09	RW/RO	Impl Spec	**Multiple Header Recording Capable (MHRC):** If this field is set to '1', indicates that the controller is capable of generating more than one error header.
08	RW/RO	0	**ECRC Check Enable (ECE):** If this field is set to '1', indicates that the ECRC checking is enabled. This field is sticky – it is neither initialized nor modified during a hot reset or FLR. If the controller does not implement the associated mechanism, then this field is cleared to '0'.
07	RO	Impl Spec	**ECRC Check Capable (ECC):** If this field is set to '1', indicates that the controller is capable of checking ECRC.
06	RW/RO	0	**ECRC Generation Enable (EGE):** If this field is set to '1', indicates that the ECRC generation is enabled. This field is sticky – it is neither initialized nor modified during a hot reset or FLR. If the controller does not implement the associated mechanism, then this field is cleared to '0'.
05	RO	Impl Spec	**ECRC Generation Capable (EGC):** If this field is set to '1', indicates that the controller is capable of generating ECRC.
04:00	RO	0	**First Error Pointer (FEP):** This field identifies the bit position of the first error reported in the AERUCES register. This field is sticky – it is neither initialized nor modified during a hot reset or FLR.

2.6.8 Offset AERCAP + 1Ch: AERHL – AER Header Log Register

This register contains the header for the TLP corresponding to a detected error. This register is sticky – it is neither initialized nor modified during a hot reset or FLR.

Byte	Type	Reset	Description
0	RO	0	**Header Byte 3 (HB3)**
1	RO	0	**Header Byte 2 (HB2)**
2	RO	0	**Header Byte 1 (HB1)**
3	RO	0	**Header Byte 0 (HB0)**
4	RO	0	**Header Byte 7 (HB7)**
5	RO	0	**Header Byte 6 (HB6)**
6	RO	0	**Header Byte 5 (HB5)**
7	RO	0	**Header Byte 4 (HB4)**
8	RO	0	**Header Byte 11 (HB11)**
9	RO	0	**Header Byte 10 (HB10)**
10	RO	0	**Header Byte 9 (HB9)**
11	RO	0	**Header Byte 8 (HB8)**
12	RO	0	**Header Byte 15 (HB15)**
13	RO	0	**Header Byte 14 (HB14)**
14	RO	0	**Header Byte 13 (HB13)**
15	RO	0	**Header Byte 12 (HB12)**

2.6.9 Offset AERCAP + 38h: AERTLP – AER TLP Prefix Log Register (Optional)

This register contains the End-End TLP prefix(es) for the TLP corresponding to a detected error. This register is sticky – it is neither initialized nor modified during a hot reset or FLR.

Byte	Type	Reset	Description
0	RO	0	**First TLP Prefix Log Byte 3 (TPL1B3)**
1	RO	0	**First TLP Prefix Log Byte 2 (TPL1B2)**
2	RO	0	**First TLP Prefix Log Byte 1 (TPL1B1)**
3	RO	0	**First TLP Prefix Log Byte 0 (TPL1B0)**
4	RO	0	**Second TLP Prefix Log Byte 3 (TPL2B3)**
5	RO	0	**Second TLP Prefix Log Byte 2 (TPL2B2)**
6	RO	0	**Second TLP Prefix Log Byte 1 (TPL2B1)**
7	RO	0	**Second TLP Prefix Log Byte 0 (TPL2B0)**
8	RO	0	**Third TLP Prefix Log Byte 3 (TPL3B3)**
9	RO	0	**Third TLP Prefix Log Byte 2 (TPL3B2)**
10	RO	0	**Third TLP Prefix Log Byte 1 (TPL3B1)**
11	RO	0	**Third TLP Prefix Log Byte 0 (TPL3B0)**
12	RO	0	**Fourth TLP Prefix Log Byte 3 (TPL4B3)**
13	RO	0	**Fourth TLP Prefix Log Byte 2 (TPL4B2)**
14	RO	0	**Fourth TLP Prefix Log Byte 1 (TPL4B1)**
15	RO	0	**Fourth TLP Prefix Log Byte 0 (TPL4B0)**

2.7 Other Capability Pointers

Though not mentioned in this specification, other capability pointers may be necessary, depending upon the implementation. Examples would be the PCI-X capability for PCI-X implementations, and potentially the vendor specific capability pointer.

These capabilities are beyond the scope of this specification.

3 Controller Registers

Controller registers are located in the MLBAR/MUBAR registers (PCI BAR0 and BAR1) that shall be mapped to a memory space that supports in-order access and variable access widths. For many computer architectures, specifying the memory space as uncacheable produces this behavior. The host shall not issue locked accesses. The host shall access registers in their native width or aligned 32-bit accesses. Violation of either of these host requirements results in undefined behavior.

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

All reserved registers and all reserved bits within registers are read-only and return 0h when read. Software shall not rely on 0h being returned.

3.1 Register Definition

The following table describes the register map for the controller.

The Vendor Specific address range starts after the last doorbell supported by the controller and continues to the end of the BAR0/1 supported range. The start of the Vendor Specific address range starts at the same location and is not dependent on the number of allocated doorbells.

Start	End	Symbol	Description
00h	07h	CAP	Controller Capabilities
08h	0Bh	VS	Version
0Ch	0Fh	INTMS	Interrupt Mask Set
10h	13h	INTMC	Interrupt Mask Clear
14h	17h	CC	Controller Configuration
18h	1Bh	Reserved	Reserved
1Ch	1Fh	CSTS	Controller Status
20h	23h	NSSR	NVM Subsystem Reset (Optional)
24h	27h	AQA	Admin Queue Attributes
28h	2Fh	ASQ	Admin Submission Queue Base Address
30h	37h	ACQ	Admin Completion Queue Base Address
38h	3Bh	CMBLOC	Controller Memory Buffer Location (Optional)
3Ch	3Fh	CMBSZ	Controller Memory Buffer Size (Optional)
40h	43h	BPINFO	Boot Partition Information (Optional)
44h	47h	BPRSEL	Boot Partition Read Select (Optional)
48h	4Fh	BPMBL	Boot Partition Memory Buffer Location (Optional)
50h	EFFh	Reserved	Reserved
F00h	FFFh	Reserved	Command Set Specific
1000h	1003h	SQ0TDBL	Submission Queue 0 Tail Doorbell (Admin)
1000h + (1 * (4 << CAP.DSTRD))	1003h + (1 * (4 << CAP.DSTRD))	CQ0HDBL	Completion Queue 0 Head Doorbell (Admin)
1000h + (2 * (4 << CAP.DSTRD))	1003h + (2 * (4 << CAP.DSTRD))	SQ1TDBL	Submission Queue 1 Tail Doorbell
1000h + (3 * (4 << CAP.DSTRD))	1003h + (3 * (4 << CAP.DSTRD))	CQ1HDBL	Completion Queue 1 Head Doorbell
1000h + (4 * (4 << CAP.DSTRD))	1003h + (4 * (4 << CAP.DSTRD))	SQ2TDBL	Submission Queue 2 Tail Doorbell
1000h + (5 * (4 << CAP.DSTRD))	1003h + (5 * (4 << CAP.DSTRD))	CQ2HDBL	Completion Queue 2 Head Doorbell
…	…	…	…
1000h+ (2y * (4 << CAP.DSTRD))	1003h + (2y * (4 << CAP.DSTRD))	SQyTDBL	Submission Queue y Tail Doorbell
1000h + ((2y + 1) * (4 << CAP.DSTRD))	1003h + ((2y + 1) * (4 << CAP.DSTRD))	CQyHDBL	Completion Queue y Head Doorbell
			Vendor Specific (Optional)

3.1.1 Offset 00h: CAP – Controller Capabilities

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

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>63:56</td><td>RO</td><td>0h</td><td>Reserved</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.</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.</td></tr>
<tr><td>47:46</td><td>RO</td><td>0h</td><td>Reserved</td></tr>
<tr><td>45</td><td>RO</td><td>Impl Spec</td><td>Boot Partition Support (BPS): This field indicates whether the controller supports Boot Partitions. If this field is set to '1', the controller supports Boot Partitions. If this field is cleared to '0', the controller does not support Boot Partitions. Refer to section 8.13.</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. A minimum of one command set shall be supported. The field is bit significant. If a bit is set to '1', then the corresponding I/O Command Set is supported. If a bit is cleared to '0', then the corresponding I/O Command Set is not supported.
<table>
<tr><th>Bit</th><th>Definition</th></tr>
<tr><td>37</td><td>NVM command set</td></tr>
<tr><td>38</td><td>Reserved</td></tr>
<tr><td>39</td><td>Reserved</td></tr>
<tr><td>40</td><td>Reserved</td></tr>
<tr><td>41</td><td>Reserved</td></tr>
<tr><td>42</td><td>Reserved</td></tr>
<tr><td>43</td><td>Reserved</td></tr>
<tr><td>44</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>36</td><td>RO</td><td>Impl Spec</td><td>NVM Subsystem Reset Supported (NSSRS): This field indicates whether the controller supports the NVM Subsystem Reset feature defined in section 7.3.1. This field is set to '1' if the controller supports the NVM Subsystem Reset feature. This field is cleared to '0' if the controller does not support the NVM Subsystem Reset feature.</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 register is 32-bits in size. This register indicates the stride between doorbell registers. The stride is specified as (2 ^ (2 + DSTRD)) in bytes. A value of 0h indicates a stride of 4 bytes, where the doorbell registers are packed without reserved space between each register. Refer to section 8.6.</td></tr>
<tr><td>31:24</td><td>RO</td><td>Impl Spec</td><td>Timeout (TO): This is the worst case time that host software shall 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.</td></tr>
<tr><td>23:19</td><td>RO</td><td>0h</td><td>Reserved</td></tr>
</table>

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>18:17</td><td>RO</td><td>Impl Spec</td><td>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 4.11 for arbitration details.

<table><tr><th>Bit</th><th>Definition</th></tr><tr><td>17</td><td>Weighted Round Robin with Urgent Priority Class</td></tr><tr><td>18</td><td>Vendor Specific</td></tr></table>
The round robin arbitration mechanism is not listed since all controllers shall support this arbitration mechanism.</td></tr>
<tr><td>16</td><td>RO</td><td>Impl Spec</td><td>Contiguous Queues Required (CQR): This field 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 field is cleared to ‘0’ if the controller supports I/O Submission Queues and I/O Completion Queues that are not physically contiguous. If this field 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’.</td></tr>
<tr><td>15:00</td><td>RO</td><td>Impl Spec</td><td>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.</td></tr>
</table>

3.1.2 Offset 08h: VS – Version

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

3.1.2.1 VS Value for 1.0 Compliant Controllers

Bit	Type	Reset	Description
31:16	RO	0001h	**Major Version Number (MJR):** Indicates the major version is “1”
15:08	RO	00h	**Minor Version Number (MNR):** Indicates the minor version is “0”.
07:00	RO	00h	Reserved

3.1.2.2 VS Value for 1.1 Compliant Controllers

Bit	Type	Reset	Description
31:16	RO	0001h	**Major Version Number (MJR):** Indicates the major version is “1”
15:08	RO	01h	**Minor Version Number (MNR):** Indicates the minor version is “1”.
07:00	RO	00h	Reserved

3.1.2.3 VS Value for 1.2 Compliant Controllers

Bit	Type	Reset	Description
31:16	RO	0001h	**Major Version Number (MJR):** Indicates the major version is “1”
15:08	RO	02h	**Minor Version Number (MNR):** Indicates the minor version is “2”.
07:00	RO	00h	Reserved

3.1.2.4 VS Value for 1.2.1 Compliant Controllers

Bit	Type	Reset	Description
31:16	RO	0001h	**Major Version Number (MJR):** Indicates the major version is “1”

15:08	RO	02h	**Minor Version Number (MNR):** Indicates the minor version is “2”.
07:00	RO	01h	**Tertiary Version Number (TER):** Indicates the tertiary version is “1”.

3.1.2.5 VS Value for 1.3 Compliant Controllers

Bit	Type	Reset	Description
31:16	RO	0001h	**Major Version Number (MJR):** Indicates the major version is “1”
15:08	RO	03h	**Minor Version Number (MNR):** Indicates the minor version is “3”.
07:00	RO	00h	**Tertiary Version Number (TER):** Indicates the tertiary version is “0”.

3.1.3 Offset 0Ch: INTMS – Interrupt Mask Set

This register 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 register when configured for MSI-X; any accesses when configured for MSI-X is undefined. For interrupt behavior requirements, refer to section 7.5.

Bit	Type	Reset	Description
31:00	RW1S	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 register). 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.

3.1.4 Offset 10h: INTMC – Interrupt Mask Clear

This register 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 register when configured for MSI-X; any accesses when configured for MSI-X is undefined. For interrupt behavior requirements, refer to section 7.5.

Bit	Type	Reset	Description
31:00	RW1C	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 register). 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.

3.1.5 Offset 14h: CC – Controller Configuration

This register modifies settings for the controller. Host software shall set the Arbitration Mechanism (CC.AMS), the Memory Page Size (CC.MPS), and the Command Set (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 I/O Submission Queue Entry Size (CC.IOSQES) should cause a controller to abort Create I/O Completion Queue or Create I/O Submission Queue commands with a status code of Invalid Queue Size.

Bit	Type	Reset	Description
31:24	RO	0	Reserved

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>23:20</td><td>RW</td><td>0</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. The required and maximum values for this field are specified in the Identify Controller data structure in Figure 109 for each I/O Command Set. The value is in bytes and is specified as a power of two (2^n).</td></tr>
<tr><td>19:16</td><td>RW</td><td>0</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. The required and maximum values for this field are specified in the Identify Controller data structure in Figure 109 for each I/O Command Set. The value is in bytes and is specified as a power of two (2^n).</td></tr>
<tr><td>15:14</td><td>RW</td><td>0h</td><td>Shutdown Notification (SHN): This field is used to initiate shutdown processing when a shutdown is occurring, (i.e., a power down condition is expected.) For a normal shutdown notification, it is expected that the controller is given time to process the shutdown notification. For an abrupt shutdown notification, the host may not wait for shutdown processing to complete before power is lost.

The 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 reboot. To determine when shutdown processing is complete, refer to CSTS.SHST. Refer to section 7.6.2 for additional shutdown processing details.

Other fields in the CC register (including the EN bit) may be modified as part of updating this field to 01b or 10b.</td></tr>
<tr><td>13:11</td><td>RW</td><td>0h</td><td>Arbitration Mechanism Selected (AMS): This field selects the arbitration mechanism to be used. This value shall only be changed when 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.

<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 – 110b</td><td>Reserved</td></tr>
<tr><td>111b</td><td>Vendor Specific</td></tr>
</table></td></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 4KB and the maximum host memory page size is 128MB. 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 EN is cleared to ‘0’.</td></tr>
<tr><td>06:04</td><td>RW</td><td>0h</td><td>I/O Command Set Selected (CSS): This field specifies the I/O Command Set that is selected for use for the I/O Submission Queues. Host software shall only select a supported I/O Command Set, as indicated in CAP.CSS. This field shall only be changed when the controller is disabled (CC.EN is cleared to ‘0’). The I/O Command Set selected shall be used for all I/O Submission Queues.

<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>NVM Command Set</td></tr>
<tr><td>001b – 111b</td><td>Reserved</td></tr>
</table></td></tr>
</table>

Bit	Type	Reset	Description
03:01	RO	0	Reserved
00	RW	0	**Enable (EN):** When set to ‘1’, then the controller shall process commands based on Submission Queue Tail doorbell writes. When cleared to ‘0’, then the controller shall not process commands nor post completion queue entries to Completion Queues. When this field transitions from ‘1’ to ‘0’, the controller is reset (referred to as a Controller Reset). The 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. The reset does not affect PCI Express registers (including MMIO MSI-X registers), nor the Admin Queue registers (AQA, ASQ, or ACQ). All other controller registers defined in this section and internal controller state (e.g., Feature values defined in section 5.21.1 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 the reset operation. Refer to section 7.3 for reset details. When this field 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 field 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 field from a ‘0’ to a ‘1’ when CSTS.RDY is a ‘1,’ or setting this field from a '1' to a '0' when CSTS.RDY is a '0,' has undefined results. The Admin Queue registers (AQA, ASQ, and ACQ) shall only be modified when EN is cleared to ‘0’.

3.1.6 Offset 1Ch: CSTS – Controller Status

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:06</td><td>RO</td><td>0</td><td>Reserved</td></tr>
<tr><td>05</td><td>RO</td><td>0</td><td>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 = ‘1’.</td></tr>
<tr><td>04</td><td>RW1C</td><td>HwInit</td><td>NVM Subsystem Reset Occurred (NSSRO): The initial value of this field is '1' if the last occurance of an NVM Subsystem Reset occured while power was applied to the NVM subsystem. The initial value of this field is '0' following an NVM Subsystem Reset due to application of power to the NVM subsystem. This field is only valid if the controller supports the NVM Subsystem Reset feature defined in section 7.3.1 as indicated by CAP.NSSRS set to ‘1’.

The reset value of this field is '0' if an NVM Subsystem Reset causes activation of a new firmware image.</td></tr>
<tr><td>03:02</td><td>RO</td><td>0</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 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>
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 reset, the behavior is undefined.</td></tr>
<tr><td>01</td><td>RO</td><td>HwInit</td><td>Controller Fatal Status (CFS): This field is set to ’1’ when a fatal controller error occurred that could not be communicated in the appropriate Completion Queue. This field is cleared to ‘0’ when a fatal controller error has not occurred. Refer to section 10.5.

The reset value of this field is '1' when a fatal controller error is detected during controller initialization.</td></tr>
<tr><td>00</td><td>RO</td><td>0</td><td>Ready (RDY): This field is set to ‘1’ when the controller is ready to accept Submission Queue Tail doorbell writes after CC.EN is set to ‘1’. This field shall be cleared to ‘0’ when CC.EN is cleared to ‘0’. Commands shall not be submitted to the controller until this field is set to ‘1’ after the CC.EN bit is set to ‘1’. Failure to follow this requirement produces undefined results. Host software shall wait a minimum of CAP.TO seconds for this field to be set to ‘1’ after setting CC.EN to ‘1’ from a previous value of ‘0’.</td></tr>
</table>

3.1.7 Offset 20h: NSSR – NVM Subsystem Reset

This optional register provides host software with the capability to initiate an NVM Subsystem Reset. Support for this register is indicated by the state of the NVM Subsystem Reset Supported (CAP.NSSRS) field. If the register is not supported, then the address range occupied by the register is reserved. Refer to section 7.3.1.

Bit	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.

3.1.8 Offset 24h: AQA – Admin Queue Attributes

This register 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 4.11. 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.

Bit	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 4.1.3. Enabling a controller while this field is cleared to 00h produces undefined results. The minimum size of the Admin Completion Queue is two entries. The maximum size of the Admin Completion Queue is 4096 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 4.1.3. Enabling a controller while this field is cleared to 00h produces undefined results. The minimum size of the Admin Submission Queue is two entries. The maximum size of the Admin Submission Queue is 4096 entries. This is a 0's based value.

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

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

Bit	Type	Reset	Description
63:12	RW	Impl Spec	**Admin Submission Queue Base (ASQB):** Indicates 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

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

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

Bit	Type	Reset	Description
63:12	RW	Impl Spec	**Admin Completion Queue Base (ACQB):** Indicates 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

3.1.11 Offset 38h: CMBLOC – Controller Memory Buffer Location

This optional register defines the location of the Controller Memory Buffer (refer to section 4.7). If CMBSZ is 0, this register is reserved.

Bit	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. This value shall be 4KB aligned.
11:03	RO	0h	Reserved
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 lower 32-bits of the address is specified. Values 0h, 2h, 3h, 4h, and 5h are valid.

3.1.12 Offset 3Ch: CMBSZ – Controller Memory Buffer Size

This optional register defines the size of the Controller Memory Buffer (refer to section 4.7). If the controller does not support the Controller Memory Buffer feature then this register shall be cleared to 0h.

Bit	Type	Reset	Description
31:12	RO	Impl Spec	**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.
11:08	RO	Impl Spec	**Size Units (SZU):** Indicates the granularity of the Size field. Value / Granularity: 0h: 4 KB 1h: 64 KB 2h: 1 MB 3h: 16 MB 4h: 256 MB 5h: 4 GB 6h: 64 GB 7h – Fh: Reserved
07:05	RO	0h	Reserved
04	RO	Impl Spec	**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 all data and metadata for commands that transfer data from the host to the controller shall be transferred from host memory.

03	RO	Impl Spec	**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 all data and metadata for commands that transfer data from the controller to the host shall be transferred to host memory.
02	RO	Impl Spec	**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 all PRP Lists and SGLs shall be placed in host memory.
01	RO	Impl Spec	**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 all Completion Queues shall be placed in host memory.
00	RO	Impl Spec	**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 all Submission Queues shall be placed in host memory.

3.1.13 Offset 40h: BPINFO – Boot Partition Information

This optional register defines the characteristics of Boot Partitions (refer to section 8.13). If the controller does not support the Boot Partitions feature then this register shall be cleared to 0h.

<table>
<tr><th>Bit</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 field 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>0h</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 8.13.

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 128KB. Both Boot Partitions are the same size.</td></tr>
</table>

3.1.14 Offset 44h: BPRSEL – Boot Partition Read Select

This optional register is used to initiate the transfer of a data in the Boot Partition (refer to section 8.13) from the controller to the host. If the controller does not support the Boot Partitions feature then this register 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.

Bit	Type	Reset	Description
31	RW	0h	**Boot Partition Identifier (BPID):** This field specifies the Boot Partition identifier for the Boot Read operation.
30	RO	0h	Reserved
29:10	RW	0h	**Boot Partition Read Offset (BPROF):** This field selects the offset into the Boot Partition, in 4KB 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 4KB to copy into the Boot Partition Memory Buffer.

3.1.15 Offset 48h: BPMBL – Boot Partition Memory Buffer Location (Optional)

This optional register specifies the memory buffer that is used as the destination for data when a Boot Partition is read (refer to section 8.13). If the controller does not support the Boot Partitions feature then this register shall be cleared to 0h.

Bit	Type	Reset	Description
63:12	RW	0h	**Boot Partition Memory Buffer Base Address (BMBBA):** Specifies the 64-bit physical address for the Boot Partition Memory Buffer. This address shall be 4KB aligned. Note that this field contains the 52 most significant bits of the 64 bit address.
11:00	RO	0h	Reserved

3.1.16 Offset (1000h + ((2y) * (4 << CAP.DSTRD))): SQyTDBL – Submission Queue y Tail Doorbell

This register defines the doorbell register that updates the Tail entry pointer for Submission Queue y. The value of y is equivalent to the Queue Identifier. This indicates to the controller that new commands have been submitted for processing.

The host should not read the doorbell registers. If a doorbell register is read, the value returned is vendor specific. Writing to a non-existent Submission Queue Tail Doorbell has undefined results.

Bit	Type	Reset	Description
31:16	RO	0	Reserved
15:00	RW	0h	**Submission Queue Tail (SQT):** Indicates the new value of the Submission Queue Tail entry pointer. This value shall overwrite any previous Submission Queue Tail entry pointer value provided. The difference between the last SQT write and the current SQT write indicates the number of commands added to the Submission Queue. Note: Submission Queue rollover needs to be accounted for.

3.1.17 Offset (1000h + ((2y + 1) * (4 << CAP.DSTRD))): CQyHDBL – Completion Queue y Head Doorbell

This register defines the doorbell register that updates the Head entry pointer for Completion Queue y. The value of y is equivalent to the Queue Identifier. This indicates Completion Queue entries that have been processed by host software.

The host should not read the doorbell registers. If a doorbell register is read, the value returned is vendor specific. Writing to a non-existent Completion Queue Head Doorbell has undefined results.

Host software should ensure it continues to process completion queue entries within Completion Queues regardless of whether there are entries available in a particular or any Submission Queue.

Bit	Type	Reset	Description
31:16	RO	0	Reserved
15:00	RW	0h	**Completion Queue Head (CQH):** Indicates the new value of the Completion Queue Head entry pointer. This value shall overwrite any previous Completion Queue Head value provided. The difference between the last CQH write and the current CQH entry pointer write indicates the number of entries that are now available for re-use by the controller in the Completion Queue. Note: Completion Queue rollover needs to be accounted for.

3.2 Index/Data Pair registers (Optional)

Index/Data Pair registers provide host software with a mechanism to access the NVM Express memory mapped registers using I/O space based registers. If supported, these registers are located in BAR2. On PC based platforms, host software (BIOS, Option ROMs, OSes) written to operate in 'real-mode' (8086 mode) are unable to access registers in a PCI Express function's address space, if the address space is memory mapped and mapped above 1MB.

The Index/Data Pair mechanism allows host software to access all of the memory mapped NVM Express registers using indirect I/O addressing in lieu of direct memory mapped access.

Note: UEFI drivers do not encounter the 1MB limitation, and thus when using EFI there is not a need for the Index/Data Pair mechanism. Thus, this feature is optional for the controller to support and may be obsoleted as UEFI becomes pervasive.

3.2.1 Restrictions

Host software shall not alternate between Index/Data Pair based access and direct memory mapped access methods. After using direct memory mapped access to the controller registers, the Index/Data Pair mechanism shall not be used.

3.2.2 Register Definition

The following registers describe the registers necessary to implement Index/Data Pair.

Start	End	Symbol	Description
00h	03h	IDX	Index register
04h	07h	DAT	Data register

3.2.3 Offset 00h: IDX – Index Register

Bit	Type	Reset	Description
31:02	RW	0h	**Index (IDX):** This register selects the Dword offset of the memory mapped NVM Express register to be accessed within the MLBAR/MUBAR registers (PCI BAR0 and BAR1). Host software shall not set this to a value beyond the maximum register offset implemented.
01:00	RO	0h	Reserved

3.2.4 Offset 04h: DAT – Data Register

Bit	Type	Reset	Description
31:00	RW	na	**Data (DAT):** This register is a “window” through which data is read or written to the memory mapped register pointed to by the Index register. A physical register is not implemented as the data is actually stored in the memory mapped registers. Since this is not a physical register, the reset value is the same as the reset value of the register that the Index register is currently pointing to.

4 Data Structures

This section describes data structures used by NVM Express.

4.1 Submission Queue & Completion Queue Definition

Sections 4.1, 4.1.1 and 4.1.2 apply to NVMe over PCIe only. For NVMe over Fabrics, refer to sections 2.4, 2.4.1 and 2.4.2 in the NVMe over Fabrics 1.0 specification.

The submitter of entries to a 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 4.1.2).

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

The consumer of entries on a 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 4.1.1).

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

Creation and deletion of Submission Queue and associated Completion Queues need to be ordered correctly by host software. Host software shall create 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 shall delete all associated Submission Queues prior to deleting a Completion Queue. To abort all commands submitted to the Submission Queue host software should issue a Delete I/O Submission Queue Command for that queue (refer to section 7.4.3).

Host software writes the Submission Queue Tail Doorbell (refer to section 3.1.16) and the Completion Queue Head Doorbell (refer to section 3.1.17) 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 register 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 should be 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. 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 4.6.

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. 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.

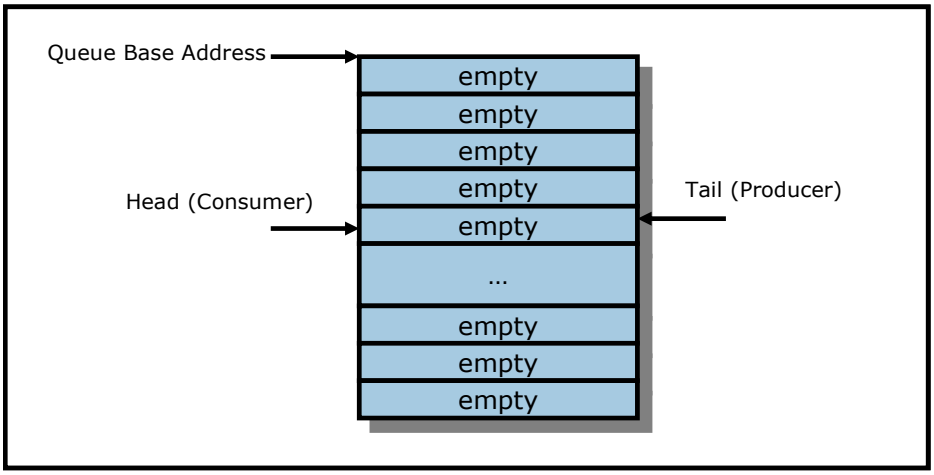
Once a Submission Queue or Completion Queue entry has been consumed, the slot in which it was placed is free and available for reuse. Altering a Submission Queue entry after that entry has been submitted but before that entry has been consumed results in undefined behavior. Altering a Completion Queue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

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 queues.

4.1.1 Empty Queue

The queue is Empty when the Head entry pointer equals the Tail entry pointer. Figure 8 defines the Empty Queue condition.

Figure 8: Empty Queue Definition

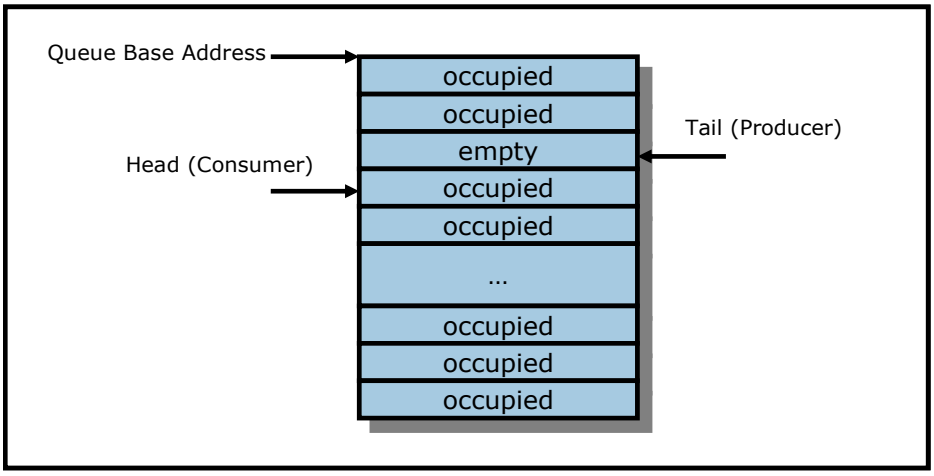


4.1.2 Full Queue

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 9 defines the Full Queue condition.

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

Figure 9: Full Queue Definition



4.1.3 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 64K 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 4K slots. One slot in each queue is not available for use due to Head and Tail entry pointer definition.

4.1.4 Queue Identifier

Each queue is identified through a 16-bit ID value that is assigned to the queue when it is created.

4.1.5 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.

4.2 Submission Queue Entry – Command Format

Each 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 NVM commands. 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 10 .

Figure 10: Command Dword 0

Bit	Description
31:16	**Command Identifier (CID):** This field specifies a unique identifier for the command when combined with the Submission Queue identifier.

<table>
<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. SGLs shall be used for all Admin and I/O commands for NVMe over Fabrics. This field shall be set to 01b for NVMe over Fabrics 1.0 implementations. The definition is described in the table below.

<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 that is byte aligned.</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 logical block 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 11.</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 4.10. 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 64 byte command format for the Admin Command Set and NVM Command Set is defined in Figure 11. 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.

Figure 11: Command Format – Admin and NVM Command Set

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>03:00</td><td>Command Dword 0 (CDW0): This field is common to all commands and is defined in Figure 10.</td></tr>
<tr><td>07:04</td><td>Namespace Identifier (NSID): This field specifies the namespace that this command applies to. If the namespace 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 this controller, unless otherwise specified.

Specifying an inactive namespace ID in a command that uses the namespace ID shall cause the controller to abort the command with status Invalid Field in Command, unless otherwise specified. Specifying an invalid namespace ID in a command that uses the namespace ID shall cause the controller to abort the command with status Invalid Namespace or Format, unless otherwise specified.</td></tr>
<tr><td>15:08</td><td>Reserved</td></tr>
<tr><td>23:16</td><td>Metadata Pointer (MPTR): This field is valid only if the command has metadata that is not interleaved with the logical block data, as specified in the Format NVM command. This is a reserved field in NVMe over Fabrics.

If CDW0.PSDT is set to 00b, then this field shall contain the address of a contiguous physical buffer of metadata and shall be Dword aligned.

If CDW0.PSDT is set to 01b, then this field shall contain the address of a contiguous physical buffer of metadata and shall be byte aligned.

If CDW0.PSDT is set to 10b, then this field shall contain the address of an SGL segment containing exactly one SGL Descriptor and shall be Qword aligned. If the SGL segment is a Data Block descriptor, then it describes the entire data transfer. Refer to section 4.4.</td></tr>
<tr><td>39:24</td><td>Data Pointer (DPTR): This field specifies the data used in the command.

If CDW0.PSDT is set to 00b, then the definition of this field is:

<table>
<tr><td>39:32</td><td>PRP Entry 2 (PRP2): This field:
a) is reserved if the data transfer does not cross a memory page boundary.
b) specifies the Page Base Address of the second memory page if the data transfer crosses exactly one memory page boundary. E.g.,:
i. 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
ii. the Offset portion of the PBAO field of PRP1 is equal to zero and the command data transfer length is greater than one memory page and less than or equal to two memory pages in size.
c) is a PRP List pointer if the data transfer crosses more than one memory page boundary. E.g.,:
i. 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
ii. 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 zero.</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 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 4.4 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>
</table>

43:40	**Command Dword 10 (CDW10):** This field is command specific Dword 10.
47:44	**Command Dword 11 (CDW11):** This field is command specific Dword 11.
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 all Admin and NVM commands, 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 12. For more details, refer to section 8.7.

Figure 12: Command Format – Admin and NVM Vendor Specific Commands (Optional)

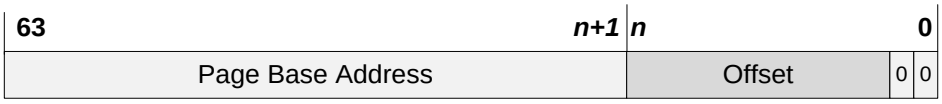
Bytes	Description
03:00	**Command Dword 0 (CDW0):** This field is common to all commands and is defined in Figure 10.
07:04	**Namespace Identifier (NSID):** This field indicates the namespace ID that this command applies to. If the namespace ID is not used for the command, then this field shall be cleared to 0h. Setting this value to FFFFFFFFh causes the command to be applied to all namespaces attached to this controller, 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 status Invalid Namespace or Format, unless otherwise specified.
15:08	Reserved
39:16	Refer to Figure 11 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.

4.3 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 13 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 13: PRP Entry Layout



The definition of a PRP entry is described in Figure 14.

Figure 14: PRP Entry – Page Base Address and Offset

Bit	Description
63:02	**Page Base Address and Offset (PBAO):** This field indicates the 64-bit physical memory page address. The lower bits (*n*:2) of this field indicate the offset within the memory page. If the memory page size is 4KB, then bits 11:02 form the Offset; if the memory page size is 8KB, then bits 12:02 form the Offset, etc. 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.
01:00	Reserved

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 15 shows the layout of a PRP List.

Figure 15: PRP List Layout

63 ***n+1***	***n*** **0**
Page Base Address *k*	0h
Page Base Address *k+1*	0h
…	
Page Base Address *k+m*	0h
Page Base Address *k+m+1*	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.

4.4 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 it 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 required by the number of logical blocks transferred.

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, then the values in the Address and Length fields of all Data Block descriptors shall have their lower two bits cleared to 00b. This requirement applies to Data Block descriptors that indicate data and/or metadata memory regions.

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 MB – 1).

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 lower 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.

Figure 16 defines the SGL segment.

Figure 16: 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 17 defines the generic SGL descriptor format.

Figure 17: 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 19)
07:04	SGL Descriptor Type (refer to Figure 18)

The SGL Descriptor Type field defined in Figure 18 specifies the SGL descriptor type. If the SGL Descriptor Type field is set to a reserved or unsupported value, then the SGL descriptor shall be processed as having

an error. If the SGL Descriptor Sub Type field is set to an unsupported value, then the descriptor shall be processed as having an SGL Descriptor Type error.

An SGL descriptor set to all zeros is an SGL Data Block descriptor with the Address field set to 00000000_00000000h and the Length field set to 00000000h may be used as a NULL descriptor.

Figure 18: SGL Descriptor Type

Code	Descriptor
0h	SGL Data Block descriptor
1h	SGL Bit Bucket descriptor
2h	SGL Segment descriptor
3h	SGL Last Segment descriptor
4h	Keyed SGL Data Block descriptor
5h – Eh	Reserved
Fh	Vendor specific

Figure 19 defines the SGL Descriptor Sub Type Values. For each Sub Type Value defined, the Descriptor Types that it applies to are indicated.

Figure 19: SGL Descriptor Sub Type Values

Descriptor Types	Sub Type Value	Sub Type Description
0h, 2h, 3h, 4h	0h	**Address:** The Address field specifies the starting 64-bit memory byte address of the Data Block, Segment, or Last Segment descriptor
0h, 2h, 3h	1h	**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 the NVMe over Fabrics specification for details on the location from which the offset is specified.
All	Ah – Fh	**NVMe Transport Specific:** The definitions for this range of Sub Types are defined by the binding section for the associated NVMe Transport.

The SGL Data Block descriptor, defined in Figure 20, describes a data block.

Figure 20: SGL Data Block descriptor

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>7:0</td><td>Address: If the SGL Identifier Descriptor Sub Type field is set 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 specified in the SGL Support field of Identify Controller then the lower two bits shall be cleared to 00b.</td></tr>
<tr><td>11:8</td><td>Length: The Length field specifies the length in bytes of the data block. A Length field set to 00000000h 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 field of Identify Controller then the lower two bits shall be 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.</td></tr>
<tr><td>14:12</td><td>Reserved</td></tr>
<tr><td>15</td><td>SGL Identifier: The definition of this field is described in the table below.

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>03:00</td><td>SGL Descriptor Sub Type field. Valid values are specified in Figure 19.</td></tr>
<tr><td>07:04</td><td>SGL Descriptor Type: 0h as specified in Figure 18.</td></tr>
</table>
</td></tr>
</table>

The SGL Bit Bucket descriptor, defined in Figure 21, is used to ignore parts of source data.

Figure 21: SGL Bit Bucket descriptor

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>7:0</td><td>Reserved</td></tr>
<tr><td>11:8</td><td>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 set to 00000000h) 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 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 memory to the controller), then the Bit Bucket Descriptor shall be treated as if the Length field were set to 00000000h (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 Number of Logical Blocks (NLB) parameter specified in NVM Command Set data transfer commands. Their length in a source data buffer is not included in the NLB parameter.</td></tr>
<tr><td>14:12</td><td>Reserved</td></tr>
<tr><td>15</td><td>SGL Identifier: The definition of this field is described in the table below.

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>03:00</td><td>SGL Descriptor Sub Type field. Valid values are specified in Figure 19.</td></tr>
<tr><td>07:04</td><td>SGL Descriptor Type: 1h as specified in Figure 18.</td></tr>
</table>
</td></tr>
</table>

The SGL Segment descriptor, defined in Figure 22, describes the next SGL segment, which is not the last SGL segment.

Figure 22: SGL Segment descriptor

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>7:0</td><td>Address: If the SGL Identifier Descriptor Sub Type field is set to 0h then the Address field specifies the starting 64-bit memory byte address of the next SGL segment, which is a 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.</td></tr>
<tr><td>11:8</td><td>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.</td></tr>
<tr><td>14:12</td><td>Reserved</td></tr>
<tr><td>15</td><td>SGL Identifier: The definition of this field is described in the table below.

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>03:00</td><td>SGL Descriptor Sub Type field. Valid values are specified in Figure 19.</td></tr>
<tr><td>07:04</td><td>SGL Descriptor Type: 2h as specified in Figure 18.</td></tr>
</table>
</td></tr>
</table>

The SGL Last Segment descriptor, defined in Figure 23, 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 23: SGL Last Segment descriptor

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>7:0</td><td>Address: If the SGL Identifier Descriptor Sub Type field is set to 0h then the Address field specifies the starting 64-bit memory byte address of the next and last SGL segment, which is a 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.</td></tr>
<tr><td>11:8</td><td>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.</td></tr>
<tr><td>14:12</td><td>Reserved</td></tr>
<tr><td>15</td><td>SGL Identifier: The definition of this field is described in the table below.

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>03:00</td><td>SGL Descriptor Sub Type field. Valid values are specified in Figure 19.</td></tr>
<tr><td>07:04</td><td>SGL Descriptor Type: 3h as specified in Figure 18.</td></tr>
</table>
</td></tr>
</table>

The Keyed SGL Data Block descriptor, defined in Figure 24, describes a keyed data block.

Figure 24: Keyed SGL Data Block descriptor

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>7:0</td><td>Address: The Address field specifies the starting 64-bit memory byte address of the data block.</td></tr>
<tr><td>10:8</td><td>Length: The Length field specifies the length in bytes of the data block. A Length field set to 000000h 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.</td></tr>
<tr><td>14:11</td><td>Key: Specifies a 32-bit key that is associated with the data block.</td></tr>
<tr><td>15</td><td>SGL Identifier: The definition of this field is described in the table below.

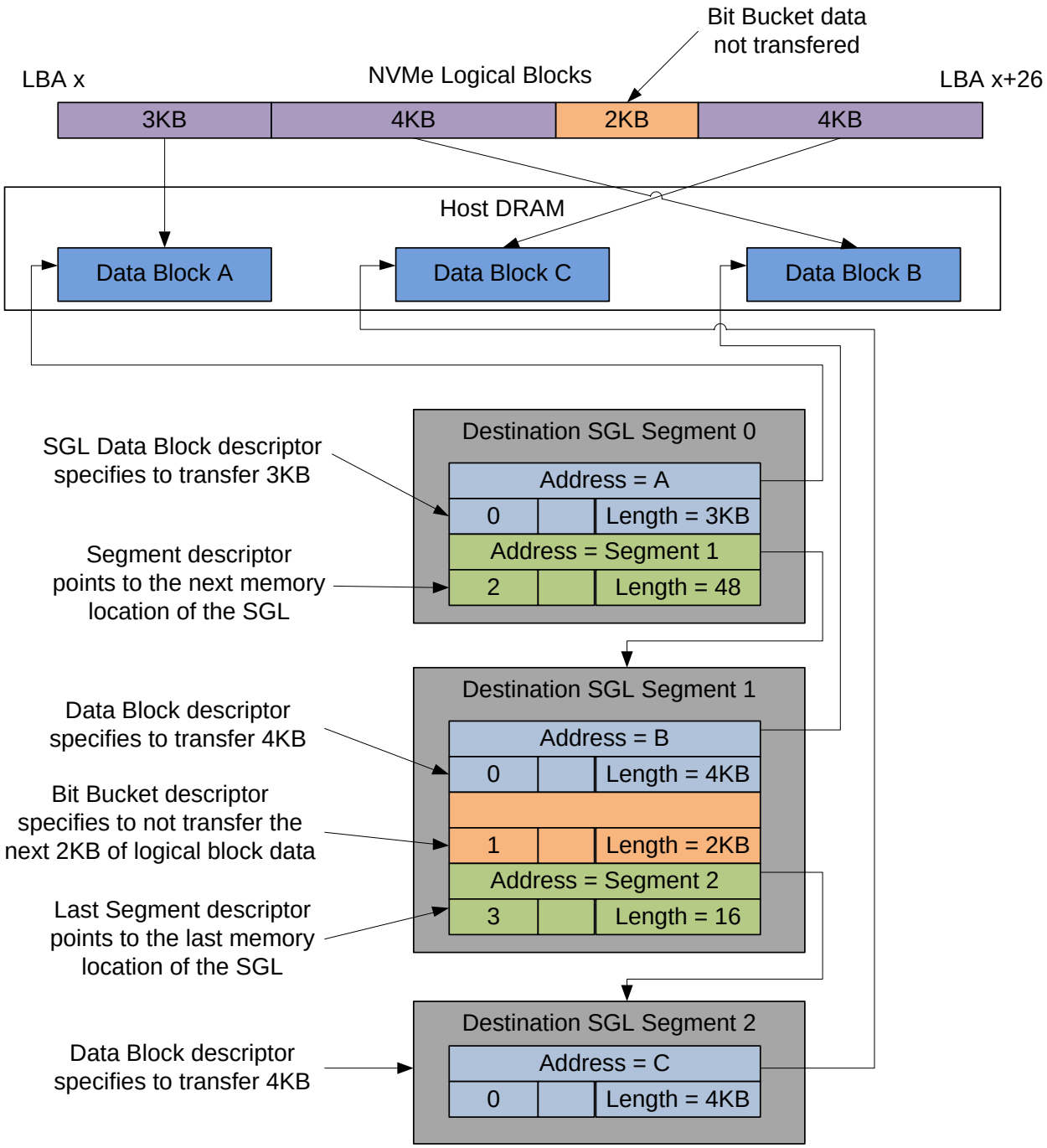
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>03:00</td><td>SGL Descriptor Sub Type field. Valid values are specified in Figure 19.</td></tr>
<tr><td>07:04</td><td>SGL Descriptor Type: 4h as specified in Figure 18.</td></tr>
</table>
</td></tr>
</table>

4.4.1 SGL Example

Figure 25 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 13KB, of which only 11KB 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 13KB. 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 KB, 4 KB and 4 KB respectively. Segment 1 of the Destination SGL contains a Bit Bucket descriptor with a length of 2 KB that specifies to not transfer (i.e., ignore) 2 KB 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 25: SGL Read Example



4.5 Metadata Region (MR)

Metadata may be supported for a namespace as either part of the logical block (creating an extended logical block which is a larger logical block that is exposed to the application) or it may be transferred as a separate buffer of data. The metadata shall not be split between the logical block and a separate metadata buffer. For writes, the metadata shall be written atomically with its associated logical block. Refer to section 8.2.

In the case where the namespace is formatted to transfer the metadata as a separate buffer of data, then the Metadata Region is used. In this case, the location of the Metadata Region is indicated by the Metadata Pointer within the command. The Metadata Pointer within the command shall be Dword aligned.

The controller may support several physical formats of logical block size and associated metadata size. There may be performance differences between different physical formats. This is indicated as part of the Identify Namespace data structure.

If the namespace is formatted to use end-to-end data protection, then the first eight bytes or last eight bytes of the metadata is used for protection information (specified as part of the NVM Format operation).

4.6 Completion Queue Entry

An entry in the Completion Queue is at least 16 bytes in size. Figure 26 describes the layout of the first 16 bytes of the Completion Queue Entry data structure. The contents of Dword 0 are command specific. If a command uses Dword 0, then the definition of this Dword is contained within the associated command definition. If a command does not use Dword 0, then the field is reserved. Dword 1 is reserved. Dword 2 is defined in Figure 27 and Dword 3 is defined in Figure 28. Any additional I/O Command Set defined in the future may use an alternate Completion Queue entry size or format.

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 26: Completion Queue Entry Layout – Admin and NVM Command Set

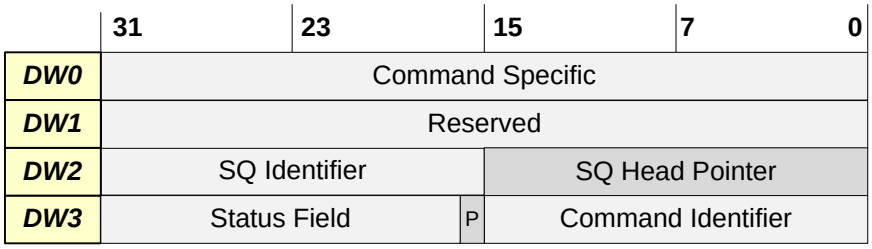


Figure 27: Completion Queue Entry: DW 2

Bit	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.
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 28: Completion Queue Entry: DW 3

Bit	Description
31:17	**Status Field (SF):** Indicates status for the command that is being completed. Refer to section 4.6.1.
16	**Phase Tag (P):** Identifies whether a Completion Queue entry is new. The Phase Tag values for all Completion Queue entries shall be initialized to '0' by host software prior to setting CC.EN to '1'. When the controller places an entry in the Completion Queue, it shall invert the Phase Tag to enable host software to discriminate a new entry. Specifically, for the first set of completion queue entries after CC.EN is set to '1' all Phase Tags are set to '1' when they are posted. For the second set of completion queue entries, when the controller has wrapped around to the top of the Completion Queue, all Phase Tags are cleared to '0' when they are posted. The value of the Phase Tag is inverted each pass through the Completion Queue. This is a reserved bit in NVMe over Fabrics.
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 at one time is 64K.

4.6.1 Status Field Definition

The Status Field defines the status for the command indicated in the completion queue entry, defined in Figure 29.

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 29: Completion Queue Entry: Status Field

Bit	Description
31	**Do Not Retry (DNR):** If set to '1', indicates that if the same command is re-submitted it 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 section 5.21.1.5), this field should be cleared to '0'. If the SCT and SC fields are cleared to 0h then this field 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 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 5.14.1.1.
29:28	Reserved
27:25	**Status Code Type (SCT):** Indicates the status code type of the completion queue entry. This indicates the type of status the controller is returning.
24:17	**Status Code (SC):** Indicates a status code identifying any error or status information for the command indicated.

4.6.1.1 Status Code Type (SCT)

Completion queue entries indicate a status code type for the type of completion being reported. Figure 30 specifies the status code type values and descriptions.

Figure 30: Status Code – Status Code Type Values

Value	Description
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.
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.
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.
3h – 6h	Reserved
7h	**Vendor Specific**

4.6.1.2 Status Code (SC)

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 – 7Fh: Applicable to Admin Command Set, or across multiple command sets.
- 80h – BFh: I/O Command Set Specific status codes.
- C0h – FFh: Vendor Specific status codes.

If there are multiple status codes that apply to a particular command failure, the controller shall report the status code with the lowest numerical value.

4.6.1.2.1 Generic Command Status Definition

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

Figure 31: Status Code – Generic Command Status Values

Value	Description
00h	**Successful Completion:** The command completed successfully.
01h	**Invalid Command Opcode:** The associated command opcode field is not valid.
02h	**Invalid Field in Command:** A reserved coded value or an unsupported value in a defined field (other than the opcode field). 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.
03h	**Command ID Conflict:** The command identifier is already in use. Note: It is implementation specific how many commands are searched for a conflict.
04h	**Data Transfer Error:** Transferring the data or metadata associated with a command had an error.
05h	**Commands Aborted due to Power Loss Notification:** Indicates that the command was aborted due to a power loss notification.
06h	**Internal Error:** The command was not completed successfully due to an internal error. Details on the internal device error are returned as an asynchronous event. Refer to section 5.2.
07h	**Command Abort Requested:** The command was aborted due to a Command Abort command being received that specified the Submission Queue Identifier and Command Identifier of this command.

Value	Description
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.
09h	**Command Aborted due to Failed Fused Command:** The command was aborted due to the other command in a fused operation failing.
0Ah	**Command Aborted due to Missing Fused Command:** The command was aborted due to the companion fused command not being found as the subsequent Submission Queue entry.
0Bh	**Invalid Namespace or Format:** The namespace or the format of that namespace is invalid.
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).
0Dh	**Invalid SGL Segment Descriptor:** The command includes an invalid SGL Last Segment or SGL Segment descriptor. This may occur when the SGL segment pointed to by an SGL Last Segment descriptor contains an SGL Segment descriptor or an SGL Last Segment descriptor or an SGL Segment descriptor. This may occur when an SGL Last Segment descriptor contains an invalid length (i.e., a length of zero or one that is not a multiple of 16).
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.
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.
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.
11h	**SGL Descriptor Type Invalid:** The type of an SGL Descriptor is a type that is not supported by the controller.
12h	**Invalid Use of Controller Memory Buffer:** The attempted use of the Controller Memory Buffer is not supported by the controller. Refer to section 4.7.
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.
14h	**Atomic Write Unit Exceeded:** The length specified exceeds the atomic write unit size.
15h	**Operation Denied:** The command was denied due to lack of access rights. Refer to the appropriate security specification (e.g., TCG SIIS). For media access commands, the Access Denied status code should be used instead.
16h	**SGL Offset Invalid:** The offset specified in a descriptor is invalid. This may occur when using capsules for data transfers in NVMe over Fabrics and an invalid offset in the capsule is specified.
17h	Reserved
18h	**Host Identifier Inconsistent Format:** The NVM subsystem detected the simultaneous use of 64-bit and 128-bit Host Identifier values on different controllers.
19h	**Keep Alive Timeout Expired:** The Keep Alive Timeout expired.
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 Keep Alive 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.
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).
1Ch	**Sanitize Failed:** The most recent sanitize operation failed and no recovery action has been successfully completed.
1Dh	**Sanitize In Progress:** The requested function (e.g., command) is prohibited while a sanitize operation is in progress. Refer to section 8.15.1.

Value	Description
1Eh	**SGL Data Block Granularity Invalid:** The Address alignment or Length granularity for an SGL Data Block descriptor is invalid. This may occur when a controller supports Dword granularity only and the lower two bits of the Address or Length are not cleared to 00b. NOTE: An implementation compliant to revision 1.2.1 or earlier may use the status code value of 15h to indicate SGL Data Block Granularity Invalid.
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: Revision 1.3 uses this status code only for Sanitize commands.
20h – 7Fh	Reserved
80h – BFh	I/O Command Set Specific
C0h – FFh	Vendor Specific

Figure 32: Status Code – Generic Command Status Values, NVM Command Set

Value	Description
80h	**LBA Out of Range:** The command references an LBA that exceeds the size of the namespace.
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 114.
82h	**Namespace Not Ready:** The namespace is not ready to be accessed. The Do Not Retry bit indicates whether re-issuing the command at a later time may succeed.
83h	**Reservation Conflict:** The command was aborted due to a conflict with a reservation held on the accessed namespace. Refer to section 8.8.
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 it is resubmitted.
85h – BFh	Reserved

4.6.1.2.2 Command Specific Errors Definition

Completion queue entries with a Status Code Type of Command Specific Errors indicate an error that is specific to a particular command opcode. Status codes of 0h to 7Fh are for Admin command errors. Status codes of 80h – BFh are specific to the selected I/O command set.

Figure 33: 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
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 Reset	Firmware Commit
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 – 6Fh	Reserved	
70h – 7Fh	Directive Specific	NOTE 1
80h – BFh	I/O Command Set Specific	NOTE 2
C0h – FFh	Vendor Specific	

NOTES:
1. The Directives Specific range defines Directives specific status values. Refer to section 9.
2. The I/O Command Set Specific range in NVMe over Fabrics defines Fabrics command specific status values.

Figure 34: Status Code – Command Specific Status Values, NVM Command Set

Value	Description	Commands Affected
80h	Conflicting Attributes	Dataset Management, Read, Write
81h	Invalid Protection Information	Compare, Read, Write, Write Zeroes
82h	Attempted Write to Read Only Range	Dataset Management, Write, Write Uncorrectable, Write Zeroes
83h - BFh	Reserved	

4.6.1.2.3 Media and Data Integrity Errors Definition

Completion queue entries with a Status Code Type 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 35: Status Code – Media and Data Integrity Error Values

Value	Description
00h – 7Fh	Reserved
80h – BFh	I/O Command Set Specific
C0h – FFh	Vendor Specific

Figure 36: Status Code – Media and Data Integrity Error Values, NVM Command Set

Value	Description
80h	**Write Fault:** The write data could not be committed to the media.
81h	**Unrecovered Read Error:** The read data could not be recovered from the media.
82h	**End-to-end Guard Check Error:** The command was aborted due to an end-to-end guard check failure.
83h	**End-to-end Application Tag Check Error:** The command was aborted due to an end-to-end application tag check failure.
84h	**End-to-end Reference Tag Check Error:** The command was aborted due to an end-to-end reference tag check failure.
85h	**Compare Failure:** The command failed due to a miscompare during a Compare command.
86h	**Access Denied:** Access to the namespace and/or LBA range is denied due to lack of access rights. Refer to the appropriate security specification (e.g., TCG SIIS).
87h	**Deallocated or Unwritten Logical Block:** The command failed due to an attempt to read from an LBA range containing a deallocated or unwritten logical block.
88h – BFh	Reserved

4.7 Controller Memory Buffer

The Controller Memory Buffer (CMB) is a region of general purpose read/write memory on the controller that 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 register.

Submission Queues in host memory require the controller to perform a PCI Express read from host memory in order to fetch the 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 initially undefined. Host software should initialize any memory before it is referenced (e.g., a Completion Queue shall be initialized by host software in order for the Phase Tag to be used correctly).

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. For a particular queue, all memory associated with it shall reside in either the Controller Memory Buffer or host memory. For all queues in the Controller Memory Buffer, the queue shall be physically contiguous.

The controller may support PRPs and SGLs in the Controller Memory Buffer. For a particular PRP List or SGL associated with a single command, all memory associated with the PRP List or SGLs shall reside in either the Controller Memory Buffer or host memory. The PRPs and SGLs for a command may only be placed in the Controller Memory Buffer if the associated command is present in a Submission Queue in the Controller Memory Buffer.

The controller may support data and metadata in the Controller Memory Buffer. All data or metadata associated with a particular command shall be located in either the Controller Memory Buffer or host memory.

If the requirements for the Controller Memory Buffer use are violated by the host, the controller shall fail the associated command with Invalid Use of Controller Memory Buffer status.

The address region allocated for the CMB shall be 4 KB aligned. It is recommended that a controller allocate the CMB on an 8 KB 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 register. The Memory Write Request to the SQ Tail doorbell register shall not have the Relaxed Ordering bit set, to ensure that it arrives at the controller after all writes to the CMB.

4.8 Namespace List

A Namespace List, defined in Figure 37, is an ordered list of namespace IDs. Unused entries are zero filled.

Figure 37: Namespace List Format

Bytes	Description
3:0	**Identifier 0:** This field contains the lowest namespace ID in the list or 0h if the list is empty.
7:4	**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.

4.9 Controller List

A Controller List, defined in Figure 38, is an ordered list of ascending controller IDs. The controller identifier is defined in bytes 79:78 of the Identify data structure in Figure 109. Unused entries are zero-filled.

Figure 38: Controller List Format

Bytes	Description
1:0	**Number of Identifiers:** This field contains the number of controller entries in the list. There may be up to 2047 identifiers in the list. A value of 0 indicates there are no controllers in the list.
3:2	**Identifier 0:** This field contains the NVM subsystem unique controller identifier for the first controller in the list, if present.
5:4	**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.

4.10 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 Identify Controller data structure in Figure 109. 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 shall be aborted. If the second command in the sequence failed, then the completion status of the first command is sequence specific.
- The LBA range, if used, shall be the same for the two commands. If the LBA ranges do not match, the commands should be aborted with status of Invalid Field in Command.
- 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. The Submission Queue Tail doorbell pointer update shall indicate both commands as part of one doorbell update.
- If the host desires to abort the fused operation, the host shall submit an Abort command separately for each of the commands.
- 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 field of Command Dword 0 in Figure 10. The Fused Operation field also indicates whether this is the first or second command in the operation.

4.11 Command Arbitration

For NVMe over PCIe, a command is submitted to the controller when a Submission Queue Tail Doorbell write by the host moves the Submission Queue Tail Pointer past the slot containing the corresponding Submission Queue entry. For NVMe over Fabrics, refer to section 1.4.14 in the NVMe over Fabrics 1.0 specification for the definition of command submission. 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.

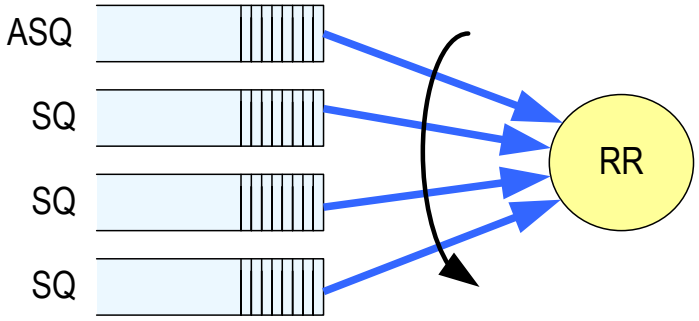
All controllers shall support the round robin command arbitration mechanism. A controller may optionally implement weighted round robin with urgent priority class and/or a vendor specific arbitration mechanism. The Arbitration Mechanism Supported field in the Controller Capabilities register (CC.AMS) 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 109), taking into consideration any latency requirements. Refer to section 5.21.1.1.

4.11.1 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 39: Round Robin Arbitration



4.11.2 Weighted Round Robin with Urgent Priority Class Arbitration

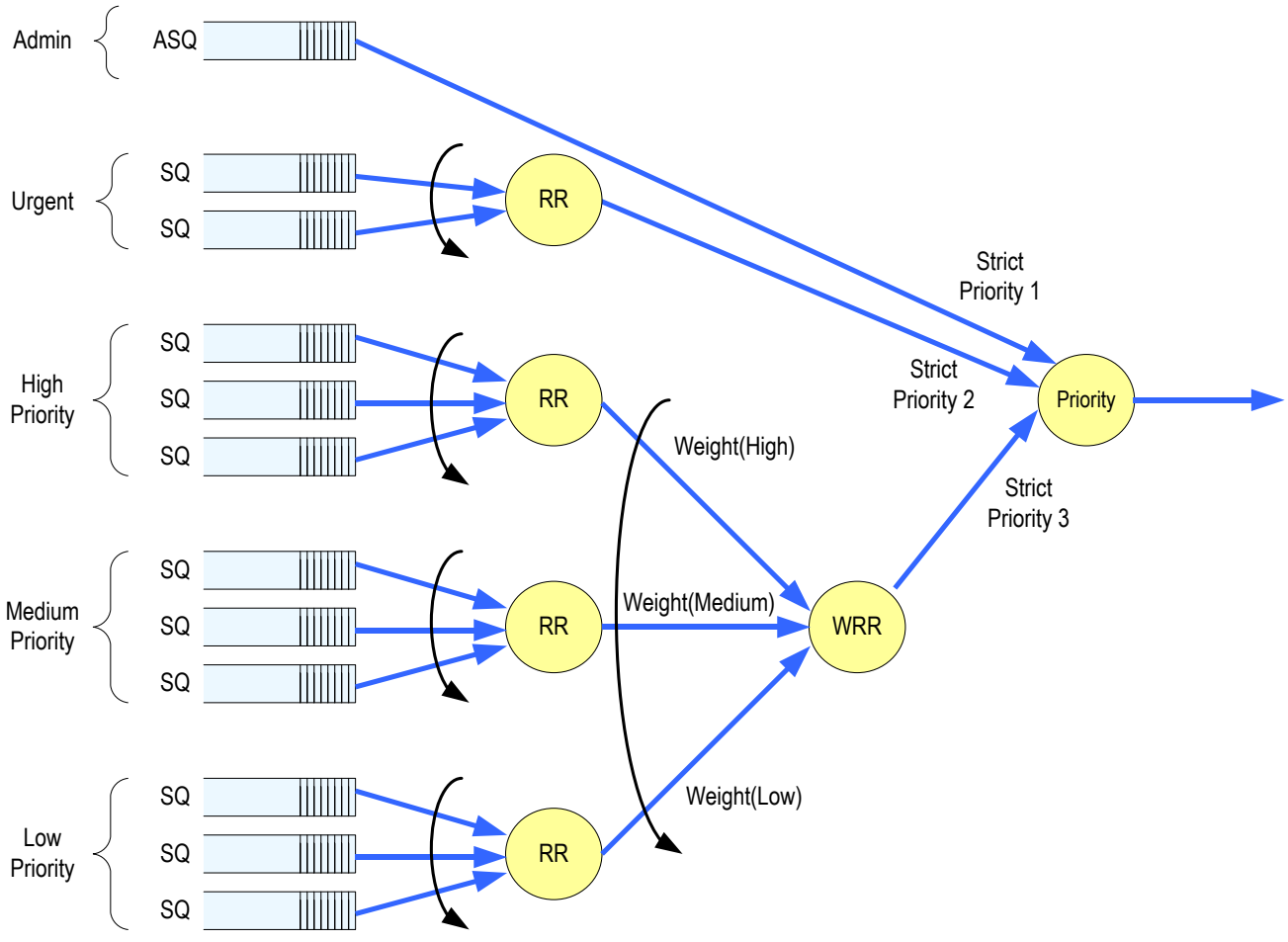
In this arbitration mechanism, there are three strict priority classes and three weighted round robin priority levels. If Submission Queue A is of higher strict priority than Submission Queue B, then all candidate commands in Submission Queue A shall start processing before candidate commands from Submission 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 Set Features. 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 40: Weighted Round Robin with Urgent Priority Class Arbitration



In Figure 40, the Priority decision point selects the highest priority candidate command selected next to start processing.

4.11.3 Vendor Specific Arbitration

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

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 4.2. The Completion Queue Entry (CQE) structure and the fields that are common to all Admin commands are defined in section 4.6. The command specific fields in the SQE and CQE structures (i.e., SQE Command Dwords 10-15 and CQE Dword 0) for the Admin Command Set are defined in this section.

For all Admin commands, Dword 14 and 15 are I/O Command Set specific.

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 41: Opcodes for Admin Commands

Opcode by Field			Combined Opcode[2]	O/M[1]	Namespace Identifier Used[3]	Command
(07)	(06:02)	(01:00)				
Generic Command	Function	Data Transfer[4]				
0b	000 00b	00b	00h	M	No	Delete I/O Submission Queue
0b	000 00b	01b	01h	M	No	Create I/O Submission Queue
0b	000 00b	10b	02h	M	Yes	Get Log Page
0b	000 01b	00b	04h	M	No	Delete I/O Completion Queue
0b	000 01b	01b	05h	M	No	Create I/O Completion Queue
0b	000 01b	10b	06h	M	Yes	Identify
0b	000 10b	00b	08h	M	No	Abort
0b	000 10b	01b	09h	M	Yes	Set Features
0b	000 10b	10b	0Ah	M	Yes	Get Features
0b	000 11b	00b	0Ch	M	No	Asynchronous Event Request
0b	000 11b	01b	0Dh	O	Yes	Namespace Management
0b	001 00b	00b	10h	O	No	Firmware Commit
0b	001 00b	01b	11h	O	No	Firmware Image Download
0b	001 01b	00b	14h	O	Yes	Device Self-test
0b	001 01b	01b	15h	O	Yes	Namespace Attachment
0b	001 10b	00b	18h	NOTE 5	No	Keep Alive
0b	001 10b	01b	19h	O	Yes	Directive Send
0b	001 10b	10b	1Ah	O	Yes	Directive Receive
0b	001 11b	00b	1Ch	O	No	Virtualization Management
0b	001 11b	01b	1Dh	O	No	NVMe-MI Send
0b	001 11b	10b	1Eh	O	No	NVMe-MI Receive
0b	111 11b	00b	7Ch	O	No	Doorbell Buffer Config
0b	111 11b	11b	7Fh	O	Refer to the NVMe over Fabrics specification.	
	I/O Command Set Specific					
1b	na	NOTE 4	80h – BFh	O		I/O Command Set specific

Opcode by Field			Combined Opcode[2]	O/M[1]	Namespace Identifier Used[3]	Command
(07)	(06:02)	(01:00)				
Generic Command	Function	Data Transfer[4]				
	Vendor Specific					
1b	na	NOTE 4	C0h – FFh	O		Vendor specific

NOTES:
1. O/M definition: O = Optional, M = Mandatory.
2. Opcodes not listed are reserved.
3. A subset of commands uses the Namespace Identifier field (CDW1.NSID). When not used, the field shall be cleared to 0h.
4. 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.
5. 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.

Figure 42 defines Admin commands that are specific to the NVM Command Set.

Figure 42: Opcodes for Admin Commands – NVM Command Set Specific

Opcode (07)	Opcode (06:02)	Opcode (01:00)	Opcode[2]	O/M[1]	Namespace Identifier Used[3]	Command
Generic Command	Function	Data Transfer[4]				
1b	000 00b	00b	80h	O	Yes	Format NVM
1b	000 00b	01b	81h	O	NOTE 5	Security Send
1b	000 00b	10b	82h	O	NOTE 5	Security Receive
1b	000 01b	00b	84h	O	No	Sanitize

NOTES:
1. O/M definition: O = Optional, M = Mandatory.
2. Opcodes not listed are reserved.
3. A subset of commands uses the Namespace Identifier field (CDW1.NSID). When not used, the field shall be cleared to 0h.
4. 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.
5. The use of the Namespace Identifier is Security Protocol specific.

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. It is implementation specific when a controller chooses to complete the Abort command when the command to abort is not found.

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 7.3.3 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 Identify Controller 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 43: Abort – Command Dword 10

Bit	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

A completion queue entry is posted to the Admin Completion Queue if the command has been completed and a corresponding completion queue entry has been posted to the appropriate Admin or I/O Completion Queue. Dword 0 of the completion queue entry indicates whether the command was aborted. If the command was successfully aborted, then bit 0 of Dword 0 is cleared to '0'. If the command was not aborted, then bit 0 of Dword 0 is set to '1'.

Command specific status values associated with the Abort command are defined in Figure 44.

Figure 44: 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, the commands are aborted.

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 Asynchronous Event Request Limit specified in the Identify Controller data structure in Figure 109.

Asynchronous events are grouped into event types. The event type information is indicated 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. An event is cleared by reading the log page associated with that event using the Get Log Page command (see section 5.14).

The following event types are defined:

- Error event: Indicates a general error that is not associated with a specific command. To clear this event, host software reads the Error Information log (refer to section 5.14.1.1) using the Get Log Page command with the Retain Asynchronous Event field cleared to '0'.
- SMART / Health Status event: Indicates a SMART or health status event. To clear this event, host software reads the SMART / Health Information log (refer to section 5.14.1.2) using Get Log Page with the Retain Asynchronous Event field 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 (see section 5.21).
- I/O Command Set events: Events that are defined by an I/O command set.
    - NVM Command Set Events:
        - Reservation Log Page Available event: Indicates that one or more Reservation Notification log pages (refer to section 5.14.1.9.1) are available. To clear this event, host software reads the Reservation Notification log page using the Get Log Page command with the Retain Asynchronous Event field cleared to '0'.
        - Sanitize Operation Completed event: Indicates that a sanitize operation has completed and status is available in the Sanitize Status log page (refer to section 5.14.1.9.2). To clear this event, host software reads the Sanitize Status log page using the Get Log Page command with the Retain Asynchronous Event field cleared to '0'.

- Vendor Specific event: Indicates a vendor specific event. To clear this event, host software reads the indicated vendor specific log page using Get Log Page command with the Retain Asynchronous Event field cleared to '0'.

Asynchronous events are reported due to a new entry being added to a log page (e.g., Error Information log) or a status update (e.g., status in the SMART / Health log). A status change may be permanent (e.g., the media has become read only) or transient (e.g., the temperature 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 the controller needs to report an event and there are no outstanding Asynchronous Event Request commands, the controller should send a single notification of that Asynchronous Event Type when an Asynchronous Event Request command 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.

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 45.

Figure 45: Status Code – Command Specific Status Values

Value	Description
5h	**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 46.

Figure 46: Asynchronous Event Request – Completion Queue Entry Dword 0

<table>
<tr><th>Bit</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 47, Figure 48, Figure 49, and Figure 50 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 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></tr>
<tr><td>0h</td><td>Error status</td></tr>
<tr><td>1h</td><td>SMART / Health status</td></tr>
<tr><td>2h</td><td>Notice</td></tr>
<tr><td>3h – 5h</td><td>Reserved</td></tr>
<tr><td>6h</td><td>I/O Command Set specific status</td></tr>
<tr><td>7h</td><td>Vendor specific</td></tr>
</table>
</td></tr>
</table>

The information in either Figure 47, Figure 48, or Figure 50 is returned in the Asynchronous Event Information field, depending on the Asynchronous Event Type.

Figure 47: Asynchronous Event Information – Error Status

Value	Description
0h	**Write to Invalid Doorbell Register:** Host software wrote the doorbell of a queue that was not created.
1h	**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.
2h	**Diagnostic Failure:** A diagnostic failure was detected. This may include a self test operation.
3h	**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 7.3.
4h	**Transient Internal Error:** A transient error occurred that is specific to a particular set of commands; controller operation may continue without a reset.
5h	**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.
6h - FFh	Reserved

Figure 48: Asynchronous Event Information – SMART / Health Status

Value	Description
0h	**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.
1h	**Temperature Threshold:** A temperature is above an over temperature threshold or below an under temperature threshold (refer to section 5.21.1.4).
2h	**Spare Below Threshold:** Available spare space has fallen below the threshold.
3h - FFh	Reserved

Figure 49: Asynchronous Event Information - Notice

Value	Description
0h	**Namespace Attribute Changed:** The Identify Namespace data structure for one or more namespaces, as well as the Namespace List returned when the Identify command is issued with the CNS field set to 02h, have changed. Host software may use this event as an indication that it should read the Identify Namespace data structures for each namespace to determine what has changed. Alternatively, host software may request the Changed Namespace List (Log Identifier 04h) to determine which namespaces in this controller have changed Identify Namespace information since the last time the log page was read. A controller shall not send this event when Namespace Utilization has changed, as this is a frequent event that does not require action by the host. 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 zero, or from a zero value to a non-zero value.
1h	**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.
2h	**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 with Retain Asynchronous Event cleared to '0' for the Telemetry Controller-Initiated Log.
3h – FFh	Reserved

Figure 50: Asynchronous Event Information – NVM Command Set Specific Status

Value	Description
0h	**Reservation Log Page Available:** Indicates that one or more Reservation Notification log pages (refer to section 5.14.1.9.1) have been added to the Reservation Notification log.
1h	**Sanitize Operation Completed:** Indicates that a sanitize operation has completed and status is available in the Sanitize Status log page (refer to section 5.14.1.9.2).
2h - FFh	Reserved

5.3 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 register. 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 51: Create I/O Completion Queue – PRP Entry 1

Bit	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 and is memory page aligned (based on the value in CC.MPS). 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 and is memory page aligned (based on the value in CC.MPS). 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 52: Create I/O Completion Queue – Command Dword 10

Bit	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 4.1.3. 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 section 3.1.17). This value shall not exceed the value reported in the Number of Queues feature (see section 5.21.1.7) 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 53: Create I/O Completion Queue – Command Dword 11

Bit	Description
31:16	**Interrupt Vector (IV):** This field indicates interrupt vector to use for this Completion Queue. This corresponds to the MSI-X or multiple message MSI vector to use. If using single message MSI or pin-based interrupts, then this field shall be cleared to 0h. In MSI-X, a maximum of 2K vectors are used. This value shall not be set to a value greater than the number of messages the controller supports (refer to MSICAP.MC.MME or MSIXCAP.MXC.TS). If the value is greater than the number of messages the controller supports, the controller should return an error of Invalid Interrupt Vector.
15:02	Reserved
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 the queue is located in the Controller Memory Buffer and PC is cleared to '0', the controller shall fail the command with Invalid Use of Controller Memory Buffer status.

5.3.1 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 54.

Figure 54: 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 is currently in use or one that is outside the range supported by the controller.
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 zero or a value which exceeds the maximum supported by the controller, specified in CAP.MQES).
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.

5.4 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 register. If a PRP List is provided to describe the SQ, 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 this 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, and Command Dword 11 fields. All other command specific fields are reserved.

Figure 55: Create I/O Submission Queue – PRP Entry 1

Bit	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 and is memory page aligned (based on the value in CC.MPS). 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 and is memory page aligned (based on the value in CC.MPS). 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 56: Create I/O Submission Queue – Command Dword 10

Bit	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 4.1.3. 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 section 3.1.16). This value shall not exceed the value reported in the Number of Queues feature (see section 5.21.1.7) 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 57: Create I/O Submission Queue – Command Dword 11

Bit	Description
31:16	**Completion Queue Identifier (CQID):** This field indicates the identifier of the Completion Queue to utilize for any command completions entries associated with this Submission Queue. The value of 0h (Admin Completion Queue) shall not be specified. If the value specified is 0h or does not correspond to a valid I/O Completion Queue, the controller should return an error of Invalid Queue Identifier.
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 4.11. **Value** / **Definition** 00b – Urgent 01b – High 10b – Medium 11b – Low
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', the controller should return an error of Invalid Field in Command. If the queue is located in the Controller Memory Buffer and PC is cleared to '0', the controller shall fail the command with Invalid Use of Controller Memory Buffer status.

5.4.1 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.

Create I/O Submission Queue command specific status values are defined in Figure 58.

Figure 58: Create I/O Submission Queue – Command Specific Status Values

Value	Description
0h	**Completion Queue Invalid:** The Completion Queue identifier specified in the command does not exist.
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 is currently in use or one that is outside the range supported by the controller.
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 zero or a value which exceeds the maximum supported by the controller, specified in CAP.MQES).

5.5 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 fail with a status value of Invalid Queue Deletion.

Note: It is not possible to delete the Admin Completion Queue.

Figure 59: Delete I/O Completion Queue – Command Dword 10

Bit	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.

5.5.1 Command Completion

A completion queue entry is posted to the Admin Completion Queue when the indicated I/O Completion Queue has been deleted. Delete I/O Completion Queue command specific status values are defined in Figure 60.

Figure 60: Delete I/O Completion 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 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.

5.6 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 61: Delete I/O Submission Queue – Command Dword 10

Bit	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.

5.6.1 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 62.

Figure 62: 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.

5.7 Doorbell Buffer Config command

The Doorbell Buffer Config command is used to provide two separate memory buffers that mirror the controller's doorbell registers defined in section 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 section 7.13 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 63: Doorbell Buffer Config – Shadow Doorbell and EventIdx

Start (Offset in Buffer)	End (Offset in Buffer)	Description
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 64: Doorbell Buffer Config – PRP Entry 1

Bit	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 63. The Shadow Doorbell buffer is updated by the host. This buffer shall be memory page aligned.

Figure 65: Doorbell Buffer Config – PRP Entry 2

Bit	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 63. The EventIdx buffer is updated by the para-virtualized controller. This buffer shall be memory page aligned.

5.7.1 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.

5.8 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 8.11). 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 66.

Figure 66: Device Self-test Namespace Test Action

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 – 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 status of Invalid Namespace or Format. If this field specifies an inactive namespace ID, then the controller shall abort the command with status 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 67: Device Self-test – Command Dword 10

<table>
<tr><th>Bit</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-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 68.

Figure 68: 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 Device Self-test command with status Device Self-test in Progress.
	2h – Extended device self-test	
	Eh – Vendor specific	Vendor specific
	Fh – Abort device self-test	If bit 0 is in the Device Self-test Options (DSTO) of the Identify Controller data structure is: a) cleared to '0', or b) set to '1' and the new Device Self-test command was received on the same controller that that the self-test operation is already in progress on, then, 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. 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. 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. 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, 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.

5.8.1 Command Completion

A completion queue entry is posted to the Admin Completion Queue after the appropriate actions are taken as specified in Figure 68. Device Self-test command specific status values are defined in Figure 69.

Figure 69: 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.

5.9 Directive Receive command

The Directive Receive command returns a data buffer that is dependent on the Directive Type. Refer to section 9.

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 70: Directive Receive – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 11 for the definition of this field.

Figure 71: Directive Receive – Command Dword 10

Bit	Description
31:00	**Number of Dwords (NUMD):** This field specifies the number of Dwords to transfer. This is a 0's based value.

Figure 72: Directive Receive – Command Dword 11

Bit	Description
31:16	**Directive Specific (DSPEC):** The interpretation of this field is Directive Type dependent. Refer to section 9.
15:08	**Directive Type (DTYPE):** This field specifies the Directive Type. Refer to Figure 288 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 9.

5.9.1 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 9.

5.10 Directive Send command

The Directive Send command transfers a data buffer that is dependent on the Directive Type to the controller. Refer to section 9.

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 73: Directive Send – Data Pointer

Bit	Description

127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 11 for the definition of this field.

Figure 74: Directive Send – Command Dword 10

Bit	Description
31:00	**Number of Dwords (NUMD):** This field specifies the number of Dwords to transfer. This is a 0's based value.

Figure 75: Directive Send – Command Dword 11

Bit	Description
31:16	**Directive Specific (DSPEC):** The interpretation of this field is Directive Type dependent. Refer to section 9.
15:08	**Directive Type (DTYPE):** This field specifies the Directive Type. Refer to Figure 288 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 9.

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 9.

5.11 Firmware Commit command

NOTE: This command was known in NVM Express revision 1.0 and 1.1 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 currently executing firmware revision may be determined from the Firmware Revision field of the Identify Controller data structure in Figure 109 or as indicated in the Firmware Slot Information log page. All controllers in the NVM subsystem share firmware image slots and the same firmware is applied to all controllers.

When modifying Boot Partitions, the host may select the Boot Partition to mark as active or replace. A Boot Partition may only be written when it is unlocked (refer to 8.13).

The Firmware Commit command uses the Command Dword 10 field. All other command specific fields are reserved.

Figure 76: Firmware Commit – Command Dword 10

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>31</td><td>Boot Partition ID (BPID): Specifies the Boot Partition that shall be used for the Commit Action, if applicable.</td></tr>
<tr><td>30:06</td><td>Reserved</td></tr>
<tr><td>05:03</td><td>Commit Action (CA): This field specifies the action that is taken 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 image specified by the Firmware Slot field. This image is not activated.</td></tr>
<tr><td>001b</td><td>Downloaded image replaces the image specified by the Firmware Slot field. This image is activated at the next reset.</td></tr>
<tr><td>010b</td><td>The image specified by the Firmware Slot field is activated at the next reset.</td></tr>
<tr><td>011b</td><td>The image specified by the Firmware Slot field is requested to be activated immediately without reset.</td></tr>
<tr><td>100-101b</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 (slot 1 – 7) to use for the operation.</td></tr>
</table>

5.11.1 Command Completion

A completion queue entry is posted to the Admin Completion Queue if the controller has completed the requested action (specified in the Commit Action field).

Requests that specify activation of a new firmware image at the next reset and return with status code value of 00h, any Controller Level Reset defined in section 7.3.2 activates the specified firmware.

Firmware Commit command specific status values are defined in Figure 77.

Figure 77: 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.
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 reset occurs prior to an NVM Subsystem Reset, the controller shall continue operation with the currently executing firmware image.
11h	**Firmware Activation Requires Reset:** The firmware commit was successful; however, the image specified does not support being activated without a reset. The image shall be activated at the next reset.
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 Identify Controller. 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 it is locked (refer to section 8.13.3).

5.12 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 copies the 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 shall ensure that image pieces do not have Dword ranges that overlap. Firmware portions may be submitted out of order to the controller. Host software shall submit image portions in order when updating a Boot Partition.

The new firmware image is not activated as part of the Firmware Image Download command. Refer to section 8.1 for details on the firmware update process. The firmware update process does not modify the contents of Boot Partitions. Refer to 8.13.2 for details on the Boot Partition update process.

Host software shall not update Boot Partitions and firmware images simultaneously. A downloaded image shall be committed using Firmware Commit before downloading another image. If the controller does not receive a Firmware Commit command, then it shall delete the portion(s) of the new image in the case of a reset.

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 78: Firmware Image Download – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the location where data should be transferred from. Refer to Figure 11 for the definition of this field.

Figure 79: Firmware Image Download – Command Dword 10

Bit	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.

Figure 80: Firmware Image Download – Command Dword 11

Bit	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.

5.12.1 Command Completion

A completion queue entry is posted to the Admin Completion Queue if this portion of the firmware image has been received by the controller. Firmware Image Download command specific status values are defined in Figure 81.

Figure 81: 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 is indicated if 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.

5.13 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 11 fields. All other command specific fields are reserved.

Figure 82: Get Features – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 11 for the definition of this field. If no data structure is used as part of this specified feature, then this field is ignored.

Figure 83: Get Features – Command Dword 10

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>31:11</td><td>Reserved</td></tr>
<tr><td>10:08</td><td>Select (SEL): This field specifies which value of the attributes to return in the provided data:

<table>
<tr><th>Select</th><th>Description</th></tr>
<tr><td>000b</td><td>Current</td></tr>
<tr><td>001b</td><td>Default</td></tr>
<tr><td>010b</td><td>Saved</td></tr>
<tr><td>011b</td><td>Supported capabilities</td></tr>
<tr><td>100b – 111b</td><td>Reserved</td></tr>
</table>

Refer to section 5.13.1 for details on the value 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 109 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 treat the Select field as though it was set to 001b (i.e., default.)</td></tr>
<tr><td>07:00</td><td>Feature Identifier (FID): This field specifies the identifier of the Feature for which to provide data.</td></tr>
</table>

Figure 84 describes the Feature Identifiers whose attributes may be retrieved using Get Features. The definition of the attributes returned and associated format is specified in the section indicated.

Figure 84: Get Features – Feature Identifiers

Description	Section Defining Format of Attributes Returned
Arbitration	Section 5.21.1.1
Power Management	Section 5.21.1.2
LBA Range Type	Section 5.21.1.3
Temperature Threshold	Section 5.21.1.4
Error Recovery	Section 5.21.1.5
Volatile Write Cache	Section 5.21.1.6
Number of Queues	Section 5.21.1.7
Interrupt Coalescing	Section 5.21.1.8
Interrupt Vector Configuration	Section 5.21.1.9
Write Atomicity	Section 5.21.1.10
Asynchronous Event Configuration	Section 5.21.1.11
Autonomous Power State Transition	Section 5.21.1.12
Host Memory Buffer	Section 5.21.1.13
Timestamp	Section 5.21.1.14
Keep Alive Timer	Section 5.21.1.15
Host Controlled Thermal Management	Section 5.21.1.16
Non-Operational Power State Config	Section 5.21.1.17
NVM Command Set Specific	
Software Progress Marker	Section 5.21.1.18
Host Identifier	Section 5.21.1.19
Reservation Notification Mask	Section 5.21.1.20
Reservation Persistence	Section 5.21.1.21

5.13.1 Select field

A Select field set 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 entry of the Get Features command.

- If Dword 0 bit 0 of the completion entry of the Get Features command is set to '1', then the Feature Identifier is saveable. If Dword 0 bit 0 of the completion entry of the Get Features command is cleared to '0', then the Feature Identifier is not saveable.
- If Dword 0 bit 1 of the completion entry of the Get Features command is set to '1', then the Feature Identifier is namespace specific and settings are applied to individual namespaces. If Dword 0 bit 1 of the completion entry of the Get Features command is cleared to '0', then the Feature Identifier is not namespace specific and its settings apply to the entire controller.
- If Dword 0 bit 2 of the completion entry of the Get Features command is set to '1', then the Feature Identifier is changeable. If Dword 0 bit 2 of the completion entry of the Get Features command is cleared to '0', then the Feature Identifier is not changeable.

5.13.2 Command Completion

A completion queue entry is posted to the Admin Completion Queue if the controller has completed returning any attributes associated with the Feature. Depending on the Feature Identifier, Dword 0 of the completion queue entry may contain feature information (refer to section 5.21.1).

5.14 Get Log Page command

The Get Log Page command returns a data buffer containing the log page requested.

The Get Log Page command uses the Data Pointer, Command Dword 10, Command Dword 11, Command Dword 12, and Command Dword 13 fields. All other command specific fields are reserved.

There are mandatory and optional Log Identifiers defined in Figure 90 and Figure 91. 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 status Invalid Field in Command.

The controller indicates if it supports 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.

Figure 85: Get Log Page – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 11 for the definition of this field.

Figure 86: Get Log Page – Command Dword 10

Bit	Description
31:16	**Number of Dwords Lower (NUMDL):** This field specifies the lower 16 bits of the number of Dwords to return. 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.
15	**Retain Asynchronous Event (RAE):** This field specifies when to retain or clear an Asynchronous Event. If this bit is cleared to '0', the corresponding Asynchronous Event is cleared after the command completes successfully. If this bit is set to '1,' the corresponding Asynchronous Event is retained (i.e., not cleared) after the command completes successfully. Host software should clear this field to '0' for log pages that are not used with Asynchronous Events. Refer to section 5.2.
14:12	Reserved
11:08	**Log Specific Field (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 87: Get Log Page – Command Dword 11

Bit	Description
31:16	Reserved
15:00	**Number of Dwords (NUMDU):** This field specifies the upper 16 bits of the number of Dwords to return.

Figure 88: Get Log Page – Command Dword 12

Bit	Description
31:00	**Log Page Offset Lower (LPOL):** The log page offset specifies the location within a log page to start returning data from. This field specifies the lower 32 bits of the log page offset. This field is Dword aligned such that the lower two bits shall be cleared to 00b.

Figure 89: Get Log Page – Command Dword 13

Bit	Description
31:00	**Log Page Offset Upper (LPOU):** This field specifies the upper 32 bits of the log page offset. Refer to the Log Page Offset Lower definition.

5.14.1 Log Specific Information

Figure 90 and Figure 91 define the Log pages that may be retrieved with the Get Log Page command.

Figure 90: Get Log Page – Log Page Identifiers

Log Identifier	O/M	Description	Reference Section
00h		Reserved	
01h	M	Error Information	5.14.1.1
02h	M	SMART / Health Information	5.14.1.2
03h	M	Firmware Slot Information	5.14.1.3
04h	O	Changed Namespace List	5.14.1.4
05h	O	Commands Supported and Effects	5.14.1.5
06h	O	Device Self-test	5.14.1.6
07h	O	Telemetry Host-Initiated	5.14.1.7
08h	O	Telemetry Controller-Initiated	5.14.1.8
09h – 6Fh		Reserved	
70h		Discovery (refer to the NVMe over Fabrics specification)	
71h – 7Fh		Reserved for NVMe over Fabrics	
80h – BFh		I/O Command Set Specific	
C0h – FFh		Vendor specific	

O/M: O = Optional, M = Mandatory

Figure 91: Get Log Page – Log Page Identifiers, NVM Command Set Specific

Log Identifier	O/M	Description	Reference Section
80h	O	Reservation Notification	5.14.1.9.1
81h	O	Sanitize Status	5.14.1.9.2
82h – BFh		Reserved	

O/M: O = Optional, M = Mandatory

5.14.1.1 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 92. The log page is a set of 64-byte entries; the maximum number of entries supported is indicated in the Identify Controller data structure in Figure 109. 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 reset.

Figure 92: Get Log Page – Error Information Log Entry (Log Identifier 01h)

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>07:00</td><td>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.</td></tr>
<tr><td>09:08</td><td>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.</td></tr>
<tr><td>11:10</td><td>Command ID: This field indicates the Command Identifier of the command that the error is assocated with. If the error is not specific to a particular command then this field shall be set to FFFFh.</td></tr>
<tr><td>13:12</td><td>Status Field: This field indicates the Status Field for the command that completed. The Status Field is located in bits 15:01, bit 00 corresponds to the Phase Tag posted for the command. If the error is not specific to a particular command then this field reports the most applicable status value.</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 first 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>10:8</td><td>Bit in command that contained the error. Valid values are 0 to 7.</td></tr>
<tr><td>7:0</td><td>Byte in command that contained the error. Valid values 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 the first LBA that experienced the error condition, if applicable.</td></tr>
<tr><td>27:24</td><td>Namespace: This field indicates 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 00h indicates that no additional information is available. Valid values are in the range of 80h to FFh.</td></tr>
<tr><td>31:29</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>63:40</td><td>Reserved</td></tr>
</table>

5.14.1.2 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. The log page shall be supported on a global basis. To request the global log page, the namespace specified is FFFFFFFFh. The log page may also be supported on a per namespace basis, as indicated in the Identify Controller data structure in Figure 109. If the log page is not supported on a per namespace basis, specifying any namespace other than FFFFFFFFh should abort the command with status Invalid Field in Command. There is not namespace specific information defined in the SMART / Health log page in this revision, thus the global 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 5.21.1.11.

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 93.

Figure 93: Get Log Page – SMART / Health Information Log

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>0</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. 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 current associated state and are not persistent.
<table>
<tr><th>Bit</th><th>Definition</th></tr>
<tr><td>0</td><td>If set to ‘1’, then the available spare space has fallen below the threshold.</td></tr>
<tr><td>1</td><td>If set to ‘1’, then a temperature is above an over temperature threshold or below an under temperature threshold (refer to section 5.21.1.4).</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>3</td><td>If set to ‘1’, then the media has been placed in read only mode.</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>7:5</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>2:1</td><td>Composite Temperature: Contains a value corresponding to a temperature in degrees Kelvin 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 5.21.1.4).

Warning and critical overheating composite temperature threshold values are reported by the WCTEMP and CCTEMP fields in the Identify Controller data structure in Figure 109.</td></tr>
<tr><td>3</td><td>Available Spare: Contains a normalized percentage (0 to 100%) of the remaining spare capacity available.</td></tr>
<tr><td>4</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%).</td></tr>
<tr><td>5</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>31:6</td><td>Reserved</td></tr>
</table>

47:32	**Data Units Read:** Contains the number of 512 byte data units the host has read from the controller; this value does not include metadata. This value is reported in thousands (i.e., a value of 1 corresponds to 1000 units of 512 bytes read) and is rounded up. When the LBA size is a value other than 512 bytes, the controller shall convert the amount of data read to 512 byte units. For the NVM command set, logical blocks read as part of Compare and Read operations shall be included in this value.
63:48	**Data Units Written:** Contains the number of 512 byte data units the host has written to the controller; this value does not include metadata. This value is reported in thousands (i.e., a value of 1 corresponds to 1000 units of 512 bytes written) and is rounded up. When the LBA size is a value other than 512 bytes, the controller shall convert the amount of data written to 512 byte units. For the NVM command set, logical blocks written as part of Write operations shall be included in this value. Write Uncorrectable commands shall not impact this value.
79:64	**Host Read Commands:** Contains the number of read commands completed by the controller. For the NVM command set, this is the number of Compare and Read commands.
95:80	**Host Write Commands:** Contains the number of write commands completed by the controller. For the NVM command set, this is the number of Write commands.
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 a shutdown notification (CC.SHN) is not received prior to loss of power.
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.
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 109. 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 the Critical Composite Temperature Threshold (CCTEMP) field in the Identify Controller data structure in Figure 109. 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 94.
203:202	**Temperature Sensor 2:** Contains the current temperature reported by temperature sensor 2. This field is defined by Figure 94.
205:204	**Temperature Sensor 3:** Contains the current temperature reported by temperature sensor 3. This field is defined by Figure 94.
207:206	**Temperature Sensor 4:** Contains the current temperature reported by temperature sensor 4. This field is defined by Figure 94.

209:208	**Temperature Sensor 5:** Contains the current temperature reported by temperature sensor 5. This field is defined by Figure 94.
211:210	**Temperature Sensor 6:** Contains the current temperature reported by temperature sensor 6. This field is defined by Figure 94.
213:212	**Temperature Sensor 7:** Contains the current temperature reported by temperature sensor 7. This field is defined by Figure 94.
215:214	**Temperature Sensor 8:** Contains the current temperature reported by temperature sensor 8. This field is defined by Figure 94.
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 8.4.5) (i.e., the Composite Temperature rose above the Thermal Management Temperature 1.) This counter shall not wrap once it reaches its maximum value. A value of zero, 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 8.4.5) (i.e., the Composite Temperature rose above the Thermal Management Temperature 2.) This counter shall not wrap once it reaches its maximum value. A value of zero, 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 8.4.5). This counter shall not wrap once it reaches its maximum value. A value of zero, 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 8.4.5). This counter shall not wrap once it reaches its maximum value. A value of zero, indicates that this transition has never occurred or this field is not implemented.
511:232	Reserved

Figure 94: Get Log Page – Temperature Sensor Data Structure

Bits	Description
15:00	**Temperature Sensor Temperature (TST):** Contains the current temperature in degrees Kelvin 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 temperature of zero degrees Kelvin. The temperature reported by a temperature sensor may be used to trigger an asynchronous event (refer to section 5.21.1.4).

5.14.1.3 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 95. This log page is global to the controller.

Figure 95: Get Log Page – Firmware Slot Information Log

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 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 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

5.14.1.4 Changed Namespace List (Log Identifier 04h)

This log page is used to describe namespaces in this controller that have changed Identify Namespace information since the last time the log page was read. The log page is a Namespace List with up to 1024 entries in it. If more than 1024 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.

5.14.1.5 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 4096 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 the I/O Command Set selected in CC.CSS).

Figure 96: Get Log Page – Commands Supported and Effects Log

Bytes	Description
03:00	**Admin Command Supported 0 (ACS0):** Contains the Commands Supported and Effects data structure (refer to Figure 97) 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 97) 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 97) 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 97) 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 97) 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 97) 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 97) 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 97) 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.

Figure 97: Get Log Page – Commands Supported and Effects Data Structure

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:19</td><td>Reserved</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>Command may be submitted when there is no other outstanding command to the same namespace and another command should not be submitted to the same namespace until this command is complete</td></tr>
<tr><td>010b</td><td>Command may be submitted when there is no other outstanding command to any namespace and another command should not be submitted to any namespace until this command is complete</td></tr>
<tr><td>011b – 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 register.</td></tr>
</table>

03	**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.
02	**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.
01	**Logical Block Content Change (LBCC):** If this bit is set to '1', then this command may modify logical block content in one or more namespaces. If this bit is cleared to '0', then this command does not modify logical block content in any namespace. Logical block content changes include a write to a logical block.
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.

5.14.1.6 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 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 98: Get Log Page – Device Self-test Log

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>0</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 – 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>1</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 set to 0h (indicating there is no device-self test operation in progress), then this field is ignored.</td></tr>
<tr><td>3:2</td><td>Reserved</td></tr>
<tr><td>31:4</td><td>Newest Self-test Result Data Structure (refer to Figure 99)</td></tr>
<tr><td>59:32</td><td>2nd newest Self-test Result Data Structure (refer to Figure 99)</td></tr>
<tr><td>…</td><td>…</td></tr>
<tr><td>535:508</td><td>19th newest Self-test Result Data Structure (refer to Figure 99)</td></tr>
<tr><td>563:536</td><td>20th newest Self-test Result Data Structure (refer to Figure 99)</td></tr>
</table>

Figure 99: Get Log Page - Self-test Result Data Structure

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>0</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 – 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 – 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>1</td><td>Segment Number: This field indicates which segment the first self-test failure occurred. The field is ignored if the Device Self-test Status field is not set to 7h.</td></tr>
<tr><td>2</td><td>
Valid Diagnostic Information: This field indicates the diagnostic failure information that is reported.

Bits 7:4 are reserved.

Bit 3 defines the SC_Valid bit. If set to ‘1’, then the contents of Status Code field is valid. If cleared to ‘0’, then the contents of Status Code field is invalid.

Bit 2 defines the SCT_Valid bit. If set to ‘1’, then the contents of Status Code Type field is valid. If cleared to ‘0’, then the contents of Status Code Type field is invalid.

Bit 1 defines the FLBA_Valid bit. If set to ‘1’, then the contents of Failing LBA field is valid. If cleared to ‘0’, then the contents of Failing LBA field is invalid.

Bit 0 defines the NSID_Valid bit. If set to ‘1’, then the contents of Namespace Identifier field is valid. If cleared to ‘0’, then the contents of Namespace Identifier field is invalid.
</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
</table>

11:4	**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 indicates the LBA of the logical block that caused the test to fail. If the device encountered more than one failed logical block during the test, then this field only indicates one of those failed logical blocks. The contents of this field are valid only when the FLBA_Valid bit is set to '1'.
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 section 4.6.1.1) 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 4.6.1.2) The contents of this field are valid only when the SC_Valid bit is set to '1'.
27:26	Vendor Specific

5.14.1.7 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 8.14). 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 when the controller processes a Get Log Page for this log with the Create Telemetry Host-Initiated Data bit set to '1' in the Log Specific 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 return an error of Invalid Field in Command. This log page is global to the controller.

Figure 100: Command Dword 10 – Log Specific Field

Bit	Description
11:09	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 subsequent Telemetry Host-Initiated Log with this bit set to '1', a Firmware Commit command, or a power on reset.

The Telemetry Host-Initiated Data consists of three areas: Telemetry Host-Initiated Data Area 1, Telemetry Host-Initiated Data Area 2, and Telemetry Host-Initiated Data Area 3. All three 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 may be calculated using the Telemetry Host-Initiated Data Area 3 Last Block field.

The controller shall return data for all blocks requested. 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 101: Get Log Page – Telemetry Host-Initiated Log (Log Identifier 07h)

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 set 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.
381:14	Reserved
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 102).
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 102).
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.

5.14.1.8 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 8.14). 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 shall persist across all 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 three areas: Telemetry Controller-Initiated Data Area 1, Telemetry Controller-Initiated Data Area 2, and Telemetry Controller-Initiated Data Area 3. All three 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 may be calculated using the Telemetry Controller-Initiated Data Area 3 Last Block field.

The controller shall return data for all blocks requested. 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 return an error of Invalid Field in Command.

Figure 102: Get Log Page – Telemetry Controller-Initiated Log (Log Identifier 08h)

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 set 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.
381:14	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 with Retain Asynchronous Event cleared to '0' for the Telemetry Controller-Initiated Log completes successfully. This value is persistent across power states and reset. 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. This field is persistent across power on.
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.

5.14.1.9 NVM Command Set Specific Log Page Identifiers

This section describes NVM Command Set Specific log pages.

5.14.1.9.1 Reservation Notification (Log Identifier 80h)

A Reservation Notification log page is created 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 a single reservation notification. The format of the log page is defined in Figure 103. This log page is global to the controller.

Figure 103: Get Log Page – Reservation Notification Log

<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 for this notification. The count starts at 0h following a controller reset, is incremented with each unique log entry, and rolls over to zero when the maximum count is reached and a log page is created. A value of 0h indicates an empty log entry.</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>0</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 zero.</td></tr>
<tr><td>1</td><td>Registration Preempted</td></tr>
<tr><td>2</td><td>Reservation Released</td></tr>
<tr><td>3</td><td>Reservation Preempted</td></tr>
<tr><td>255:4</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 zero 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>

5.14.1.9.2 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 8.15). The Get Log Page command returns a data buffer containing a log page formatted as defined in Figure 104. This log page is global to the NVM subsystem and 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 field in Identify Controller is not cleared to zero (i.e., the Sanitize command is supported), then this log page shall be supported. If the Sanitize Capabilities field in Identify Controller is cleared to zero, then this log page is reserved.

Figure 104: Get Log Page – Sanitize Status Log

<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 the SSTAT field is not set to 010b.</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 non-volatile storage in the NVM subsystem has not been written to:
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 non-volatile storage in the NVM subsystem has been written to:
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 00000b 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.</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-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 178.</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). 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). 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>19:16</td><td>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). 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>511:20</td><td>Reserved</td></tr>
</table>

5.14.2 Command Completion

A completion queue entry is posted to the Admin Completion Queue if the log has been transferred to the memory buffer indicated in PRP Entry 1. Get Log Page command specific status values are defined in Figure 105.

Figure 105: Get Log Page – Command Specific Status Values

Value	Description
9h	**Invalid Log Page:** The log page indicated is invalid. This error condition is also returned if a reserved log page is requested.

5.15 Identify command

The Identify command returns a data buffer that describes information about the NVM subsystem, the controller or the namespace(s). The data structure is 4096 bytes in size.

The data structure returned, defined in Figure 106, is based on the Controller or Namespace Structure (CNS) field. If there are fewer entries to return for the data structure indicated based on CNS value, then the unused portion of the list is zero filled. If a controller does not support a CNS value then it shall abort the command with a status of Invalid Field in Command.

NOTE: The CNS field was specified as a one bit field in revision 1.0 and as 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 1.1, respectively, are indeterminate.

The Identify Controller data structure and Identify Namespace data structure include several identifiers. The format and layout of these identifiers is described in section 7.10.

Figure 106: Identify – Data Structure Returned

CNS Value	O/M	Definition
00h	M	The Identify Namespace data structure is returned to the host for the namespace specified in the Namespace Identifier (CDW1.NSID) field if it is an active NSID. If the specified namespace is not an active NSID, then the controller returns a zero filled data structure. If the controller supports Namespace Management and CDW1.NSID is set to FFFFFFFFh, the controller returns an Identify Namespace data structure that specifies capabilities that are common across namespaces for this controller. If the controller does not support Namespace Management and CDW1.NSID is set to FFFFFFFFh, the controller shall fail the command with a status code of Invalid Namespace or Format.
01h	M	The Identify Controller data structure is returned to the host for this controller.

02h	M	A list of 1024 namespace IDs is returned containing active NSIDs in increasing order that are greater than the value specified in the Namespace Identifier (CDW1.NSID) field of the command. The controller should abort the command with status code Invalid Namespace or Format if CDW1.NSID is set to FFFFFFFEh or FFFFFFFFh. Note that CDW1.NSID 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 4.8).
03h	M	A list of Namespace Identification Descriptor structures (refer to Figure 116) is returned to the host for the namespace specified in the Namespace Identifier (CDW1.NSID) field if it is an active NSID. The controller may return any number of variable length Namespace Identification Descriptor structures that fit into the 4096 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. If the hosts sees an unknown descriptor type it should continue parsing the structure. A controller shall not return multiple descriptors with the same Namespace Identification Descriptor Type (NIDT). A controller shall return at least one descriptor identifying the namespace.
04h – 0Fh		Reserved
Controller and Namespace Management		
10h	NOTE 1	A list of up to 1024 namespace IDs is returned to the host containing allocated NSIDs with a namespace identifier greater than the value specified in the Namespace Identifier (CDW1.NSID) field. The controller should abort the command with status code Invalid Namespace or Format if CDW1.NSID is set to FFFFFFFEh or FFFFFFFFh. Note that CDW1.NSID may be cleared to 0h to retrieve a Namespace List including the namespace starting with NSID of 1h.
11h	NOTE 1	The Identify Namespace data structure is returned to the host for the namespace specified in the Namespace Identifier (CDW1.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 an invalid NSID then the controller shall fail the command with a status code of Invalid Namespace or Format. If CDW1.NSID is set to FFFFFFFFh then the controller should fail the command with a status code of Invalid Namespace or Format.
12h	NOTE 1	A Controller List of up to 2047 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 that are attached to the namespace specified in the Namespace Identifier (CDW1.NSID) field.
13h	NOTE 1	A Controller List of up to 2047 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 in the NVM subsystem that may or may not be attached to namespace(s).
14h	NOTE 2	The Primary Controller Capabilities Structure (refer to Figure 110) is returned to the host for the primary controller specified.
15h	NOTE 2	A Secondary Controller List (refer to Figure 111) is returned to the host for up to 127 secondary controllers associated with the primary controller issuing this command. The list contains entries for controller identifiers greater than or equal to the value specified in the Controller Identifier (CDW10.CNTID) field.
16h – 1Fh		Reserved
Future Definition		
20h – FFh		Reserved
NOTES: 1. Mandatory for controllers that support Namespace Management. 2. Mandatory for controllers that support Virtualization Enhancements.		

The Identify command uses the Data Pointer and Command Dword 10 fields. All other command specific fields are reserved.

Figure 107: Identify – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 11 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 108: Identify – Command Dword 10

Bit	Description
31:16	**Controller Identifier (CNTID):** This field specifies the controller identifier used as part of some Identify operations. If the 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). Controllers that support Namespace Management shall support this field. This field is used for Identify operations with a CNS value of 12h or 13h. This field should be cleared to 0h for Identify operations with a CNS value of 00h, 01h, 02h, 10h, and 11h.
15:08	Reserved
07:00	**Controller or Namespace Structure (CNS):** This field specifies the information to be returned to the host. Refer to Figure 106.

Figure 109: Identify – Identify Controller Data Structure

Bytes	O/M	Description
Controller Capabilities and Features		
01:00	M	**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 section 2.1.1.
03:02	M	**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 section 2.1.17.
23:04	M	**Serial Number (SN):** Contains the serial number for the NVM subsystem that is assigned by the vendor as an ASCII string. Refer to section 7.10 for unique identifier requirements. Refer to section 1.5 for ASCII string requirements.
63:24	M	**Model Number (MN):** Contains the model number for the NVM subsystem that is assigned by the vendor as an ASCII string. Refer to section 7.10 for unique identifier requirements. Refer to section 1.5 for ASCII string requirements.
71:64	M	**Firmware Revision (FR):** Contains the currently active firmware revision for the NVM subsystem. This is the same revision information that may be retrieved with the Get Log Page command, refer to section 5.14.1.3. Refer to section 1.5 for ASCII string requirements.
72	M	**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 4.11.
75:73	M	**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.

Bytes	O/M	Description
76	O	**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 7:3 are reserved 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. Bit 1: If set to '1' then the NVM subsystem may be connected to more than one host (e.g., it may contain two or more controllers). If cleared to '0' then the NVM subsystem may only be connected to a single host (e.g., it contains only a single controller). 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.
77	M	**Maximum Data Transfer Size (MDTS):** This field indicates the maximum data transfer size between the host and the controller. The host should not submit a command that exceeds this transfer size. If a command is submitted that exceeds the transfer size, then the command is aborted with a status 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 no restrictions on transfer size. The restriction includes metadata if it is interleaved with the logical block data. The restriction does not apply to commands that do not transfer data between the host and the controller (e.g., Write Uncorrectable command or Write Zeroes command). 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	**Controller ID (CNTLID):** Contains the NVM subsystem unique controller identifier associated with the controller. Refer to section 7.10 for unique identifier requirements.
83:80	M	**Version (VER):** This field contains the value reported in the Version register defined in section 3.1.2. Implementations compliant to revision 1.2 or later of this specification shall report a non-zero value in this field.
87:84	M	**RTD3 Resume Latency (RTD3R):** This field indicates the typical latency in microseconds resuming from Runtime D3 (RTD3). Refer to section 8.4.4 for test conditions. A value of 0h indicates RTD3 Resume Latency is not reported.
91:88	M	**RTD3 Entry Latency (RTD3E):** This field indicates the typical latency in microseconds to enter Runtime D3 (RTD3). Refer to section 8.4.4 for test conditions. A value of 0h indicates RTD3 Entry Latency is not reported.
95:92	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. Bits 31:10 are reserved. Bit 9 is set to '1' if the controller supports sending Firmware Activation Notices. 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 sending Namespace Attribute Notices 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.

Bytes	O/M	Description
99:96	M	**Controller Attributes (CTRATT):** This field indicates attributes of the controller. Bits 31:2 are reserved. Bit 1 (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 5.21.1.17. Bit 0 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.
111:100		Reserved
127:112	O	**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 (NVMe-MI) 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. See the IEEE EUI-64 guidelines for more information. This field is big endian (refer to section 7.10). When not implemented, this field contains a value of 0h.
239:128		Reserved
255:240		Refer to the NVMe Management Interface Specification for definition.
Admin Command Set Attributes & Optional Controller Capabilities		

Bytes	O/M	Description
257:256	M	**Optional Admin Command Support (OACS):** This field indicates the optional Admin commands and features supported by the controller. Refer to section 5. Bits 15:9 are reserved. 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 9. 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 and Namespace Attachment commands. If cleared to '0' then the controller does not support the Namespace Management and Namespace Attachment commands. 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	**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	**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 (see 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.

Bytes	O/M	Description
260	M	**Firmware Updates (FRMW):** This field indicates capabilities regarding firmware updates. Refer to section 8.1 for more information on the firmware update process. Bits 7:5 are reserved. Bit 4 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. Bits 3:1 indicate the number of firmware slots that the controller supports. This field shall specify a value between one and seven, indicating that at least one firmware slot is supported and up to seven maximum. This corresponds to firmware slots 1 through 7. Bit 0 if set to '1' indicates that the first firmware slot (slot 1) is read only. If cleared to '0' then the first firmware slot (slot 1) is read/write. Implementations may choose to have a baseline read only firmware image.
261	M	**Log Page Attributes (LPA):** This field indicates optional attributes for log pages that are accessed via the Get Log Page command. Bits 7:4 are reserved. Bit 3 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. Bit 2 if set to '1' then the controller supports extended data for Get Log Page (including extended Number of Dwords and Log Page Offset fields). Bit 2 if cleared to '0' then the controller does not support extended data for Get Log Page. Bit 1 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. Bit 0 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.
262	M	**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.
263	M	**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 8.4. 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).
264	M	**Admin Vendor Specific Command Configuration (AVSCC):** This field indicates the configuration settings for Admin Vendor Specific command handling. Refer to section 8.7. Bits 7:1 are reserved. Bit 0 if set to '1' indicates that all Admin Vendor Specific Commands use the format defined in Figure 12. If cleared to '0' indicates that the format of all Admin Vendor Specific Commands are vendor specific.

Bytes	O/M	Description
265	O	**Autonomous Power State Transition Attributes (APSTA):** This field indicates the attributes of the autonomous power state transition feature. Refer to section 8.4.2. 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	**Warning Composite Temperature Threshold (WCTEMP):** This field indicates the minimum Composite Temperature field value (reported in the SMART / Health Information log in Figure 93) that indicates an overheating condition during which controller operation continues. Immediate remediation is recommended (e.g., additional cooling or workload reduction). The platfom should strive to maintain a composite temperature below this value. A value of 0h in this field indicates that no warning temperature threshold value is reported by the controller. Implementations compliant to revision 1.2 or later of this specification 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	**Critical Composite Temperature Threshold (CCTEMP):** This field indicates the minimum Composite Temperature field value (reported in the SMART / Health Information log in Figure 93) that indicates a critical overheating condition (e.g., may prevent continued normal operation, possibility of data loss, automatic device shutdown, extreme peformance 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 to revision 1.2 or later of this specification shall report a non-zero value in this field.
271:270	O	**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	**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 4KB units. This value shall be larger 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	**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 4KB 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	**Total NVM Capacity (TNVMCAP):** This field indicates the total NVM capacity in the NVM subsystem. The value is in bytes. This field shall be supported if Namespace Management and Namespace Attachment commands are supported.
311:296	O	**Unallocated NVM Capacity (UNVMCAP):** This field indicates the unallocated NVM capacity in the NVM subsystem. The value is in bytes. This field shall be supported if Namespace Management and Namespace Attachment commands are supported.

<table>
<tr><th>Bytes</th><th>O/M</th><th>Description</th></tr>
<tr><td>315:312</td><td>O</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 8.10.

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:24</td><td>Access Size: This field indicates the size that may be read or written per RPMB access by Security Send or Security Receive commands for this controller in 512B units. This is a 0’s based value. A value of 0h indicates a size of 512B.</td></tr>
<tr><td>23:16</td><td>Total Size: This field indicates the size of each RPMB supported in the controller in 128KB units. This is a 0’s based value. A value of 0h indicates a size of 128KB.</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</td></tr>
<tr><td>001b-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>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>Device Self-test Options (DSTO): This field indicates the optional Device Self-test command or operation behaviors supported by the controller or NVM subsystem.

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>
<tr><td>319</td><td>M</td><td>Firmware Update Granularity (FWUG): This field indicates the minimum granularity and alignment of the data provided in the Firmware Image Download command. If the data in the Firmware Image Download command does not conform to these granularity and alignment requirements, the firmware update may fail. For the broadest interoperability with software, it is recommended that the controller set this value to the lowest value possible.

The value is reported in 4KB units (1h corresponds to 4KB, 2h corresponds to 8KB, etc.). 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).</td></tr>
<tr><td>321:320</td><td>M</td><td>Keep Alive Support (KAS): This field indicates the granularity of the Keep Alive Timer in 100 ms units (refer to section 7.12). If this field is cleared to 0h then Keep Alive is not supported. Keep Alive shall be supported for NVMe over Fabrics implementations.</td></tr>
</table>

Bytes	O/M	Description
323:322	O	**Host Controlled Thermal Management Attributes (HCTMA):** This field indicates the attributes of the host controlled thermal management feature. Refer to section 8.4.5. 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	**Minimum Thermal Management Temperature (MNTMT):** This field indicates the minimum temperature, in degrees Kelvin, 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 0000h indicates that the controller does not report this field or the host controlled thermal management feature (refer to section 8.4.5) is not supported.
327:326	O	**Maximum Thermal Management Temperature (MXTMT):** This field indicates the maximum temperature, in degrees Kelvin, 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 0000h indicates that the controller does not report this field or the host controlled thermal management feature is not supported.
331:328	O	**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 is reserved. Refer to section 8.15. Bits 31:3 are reserved. Bit 2 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. Bit 1 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. Bit 0 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.
511:332		Reserved
NVM Command Set Attributes		
512	M	**Submission Queue Entry Size (SQES):** This field defines the required and maximum Submission Queue entry size when using the NVM Command Set. Bits 7:4 define the maximum Submission Queue entry size when using the NVM Command Set. This value is larger than or equal to the required SQ entry size. The value is in bytes and is reported as a power of two (2^*n*). The recommended value is 6, corresponding to a standard NVM Command Set SQ entry size of 64 bytes. Controllers that implement proprietary extensions may support a larger value. Bits 3:0 define the required Submission Queue Entry size when using the NVM Command Set. 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.

Bytes	O/M	Description
513	M	**Completion Queue Entry Size (CQES):** This field defines the required and maximum Completion Queue entry size when using the NVM Command Set. Bits 7:4 define the maximum Completion Queue entry size when using the NVM Command Set. This value is larger than or equal to the required CQ entry size. The value is in bytes and is reported as a power of two (2^*n*). The recommended value is 4, corresponding to a standard NVM Command Set CQ entry size of 16 bytes. Controllers that implement proprietary extensions may support a larger value. Bits 3:0 define the required Completion Queue entry size when using the NVM Command Set. 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	**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 and optional for NVMe over PCIe implementations. If the field is not used, it shall be cleared to 0h.
519:516	M	**Number of Namespaces (NN):** This field defines the maximum number of namespaces supported by the controller. This field also represents the maximum value of a valid NSID for the controller.
521:520	M	**Optional NVM Command Support (ONCS):** This field indicates the optional NVM commands and features supported by the controller. Refer to section 6. Bits 15:7 are reserved. 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 5.21.1.14. 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 it shall support the following commands associated with reservations: Reservation Report, Reservation Register, Reservation Acquire, and Reservation Release. Refer to section 8.8 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 Write Zeroes command. If cleared to '0' then the controller does not support the Write Zeroes command. Bit 2 if set to '1' then the controller supports the Dataset Management command. If cleared to '0' then the controller does not support the Dataset Management command. Bit 1 if set to '1' then the controller supports the Write Uncorrectable command. If cleared to '0' then the controller does not support the Write Uncorrectable command. Bit 0 if set to '1' then the controller supports the Compare command. If cleared to '0' then the controller does not support the Compare command.

Bytes	O/M	Description
523:522	M	**Fused Operation Support (FUSES):** This field indicates the fused operations that the controller supports. Refer to section 6.2. Bits 15:1 are reserved. Bit 0 if set to '1' then the controller supports the Compare and Write fused operation. If cleared to '0' then the controller does not support the Compare and Write fused operation. Compare shall be the first command in the sequence.
524	M	**Format NVM Attributes (FNA):** This field indicates attributes for the Format NVM command. Bits 7:3 are reserved. 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 an 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. Bit 0 indicates whether the format operation (excluding secure erase) applies to all namespaces in an NVM subsystem or is specific to a particular namespace. If set to '1', then all namespaces in an 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 an NVM subsystem. If cleared to '0', then the controller supports format on a per namespace basis.
525	M	**Volatile Write Cache (VWC):** This field indicates attributes related to the presence of a volatile write cache in the implementation. Bits 7:1 are reserved. 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 may issue Flush commands and control whether the volatile write cache is enabled with Set Features specifying the Volatile Write Cache feature identifier. If a volatile write cache is not present, Flush commands complete successfully and have no effect, Set Features with the Volatile Write Cache identifier field set shall fail with Invalid Field status, and Get Features with the Volatile Write Cache identifier field set should fail with Invalid Field status.

Bytes	O/M	Description
527:526	M	**Atomic Write Unit Normal (AWUN):** This field indicates the size of the write operation guaranteed to be written atomically to the NVM across all namespaces with any supported namespace format during normal operation. This field is specified in logical blocks and is a 0's based value. If a specific namespace guarantees a larger size than is reported in this field, then this namespace specific size is reported in the NAWUN field in the Identify Namespace data structure. Refer to section 6.4. If a write command is submitted with size less than or equal to the AWUN value, the host is guaranteed that the write command is atomic to the NVM with respect to other read or write commands. If a write command is submitted with size greater than the AWUN value, then there is no guarantee of command atomicity. AWUN does not have any applicability to write errors caused by power failure (refer to Atomic Write Unit Power Fail). A value of FFFFh indicates all commands are atomic as this is the largest command size. It is recommended that implementations support a minimum of 128KB (appropriately scaled based on LBA size).
529:528	M	**Atomic Write Unit Power Fail (AWUPF):** This field indicates the size of the write operation guaranteed to be written atomically to the NVM across all namespaces with any supported namespace format during a power fail or error condition. If a specific namespace guarantees a larger size than is reported in this field, then this namespace specific size is reported in the NAWUPF field in the Identify Namespace data structure. Refer to section 6.4. This field is specified in logical blocks and is a 0's based value. The AWUPF value shall be less than or equal to the AWUN value. If a write command is submitted with size less than or equal to the AWUPF value, the host is guaranteed that the write is atomic to the NVM with respect to other read or write commands. If a write command is submitted that is greater than this size, there is no guarantee of command atomicity. If the write size is less than or equal to the AWUPF value and the write command fails, then subsequent read commands for the associated logical blocks shall return data from the previous successful write command. If a write command is submitted with size greater than the AWUPF value, then there is no guarantee of data returned on subsequent reads of the associated logical blocks.
530	M	**NVM Vendor Specific Command Configuration (NVSCC):** This field indicates the configuration settings for NVM Vendor Specific command handling. Refer to section 8.7. Bits 7:1 are reserved. Bit 0 if set to '1' indicates that all NVM Vendor Specific Commands use the format defined in Figure 12. If cleared to '0' indicates that the format of all NVM Vendor Specific Commands are vendor specific.
531	M	Reserved

<table>
<tr><th>Bytes</th><th>O/M</th><th>Description</th></tr>
<tr><td>533:532</td><td>O</td><td>Atomic Compare & Write Unit (ACWU): This field indicates the size of the write operation guaranteed to be written atomically to the NVM across all namespaces with any supported namespace format for a Compare and Write fused operation.

If a specific namespace guarantees a larger size than is reported in this field, then this namespace specific size is reported in the NACWU field in the Identify Namespace data structure. Refer to section 6.4.

This field shall be supported if the Compare and Write fused command is supported. This field is specified in logical blocks and is a 0's based value. If a Compare and Write is submitted that requests a transfer size larger than this value, then the controller may fail the command with a status code of Invalid Field in Command. If Compare and Write is not a supported fused command, then this field shall be 0h.</td></tr>
<tr><td>535:534</td><td>M</td><td>Reserved</td></tr>
<tr><td>539:536</td><td>O</td><td>SGL Support (SGLS): This field indicates if SGLs are supported for the NVM Command Set and the particular SGL types supported. Refer to section 4.4.

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:21</td><td>Reserved</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>If set to '1', then use of a byte aligned contiguous physical buffer of metadata (the Metadata Pointer field in Figure 11) 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>15:03</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 for the NVM Command Set. 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 4.4).</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>767:540</td><td></td><td>Reserved</td></tr>
<tr><td>1023:768</td><td>M</td><td>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 7.9 for the definition of NVMe Qualified Name.</td></tr>
<tr><td>1791:1024</td><td></td><td>Reserved</td></tr>
</table>

Bytes	O/M	Description
2047:1792		Refer to the NVMe over Fabrics specification.
Power State Descriptors		
2079:2048	M	**Power State 0 Descriptor (PSD0):** This field indicates the characteristics of power state 0. The format of this field is defined in Figure 113.
2111:2080	O	**Power State 1 Descriptor (PSD1):** This field indicates the characteristics of power state 1. The format of this field is defined in Figure 113.
2143:2112	O	**Power State 2 Descriptor (PSD2):** This field indicates the characteristics of power state 2. The format of this field is defined in Figure 113.
2175:2144	O	**Power State 3 Descriptor (PSD3):** This field indicates the characteristics of power state 3. The format of this field is defined in Figure 113.
2207:2176	O	**Power State 4 Descriptor (PSD4):** This field indicates the characteristics of power state 4. The format of this field is defined in Figure 113.
2239:2208	O	**Power State 5 Descriptor (PSD5):** This field indicates the characteristics of power state 5. The format of this field is defined in Figure 113.
2271:2240	O	**Power State 6 Descriptor (PSD6):** This field indicates the characteristics of power state 6. The format of this field is defined in Figure 113.
2303:2272	O	**Power State 7 Descriptor (PSD7):** This field indicates the characteristics of power state 7. The format of this field is defined in Figure 113.
2335:2304	O	**Power State 8 Descriptor (PSD8):** This field indicates the characteristics of power state 8. The format of this field is defined in Figure 113.
2367:2336	O	**Power State 9 Descriptor (PSD9):** This field indicates the characteristics of power state 9. The format of this field is defined in Figure 113.
2399:2368	O	**Power State 10 Descriptor (PSD10):** This field indicates the characteristics of power state 10. The format of this field is defined in Figure 113.
2431:2400	O	**Power State 11 Descriptor (PSD11):** This field indicates the characteristics of power state 11. The format of this field is defined in Figure 113.
2463:2432	O	**Power State 12 Descriptor (PSD12):** This field indicates the characteristics of power state 12. The format of this field is defined in Figure 113.
2495:2464	O	**Power State 13 Descriptor (PSD13):** This field indicates the characteristics of power state 13. The format of this field is defined in Figure 113.
2527:2496	O	**Power State 14 Descriptor (PSD14):** This field indicates the characteristics of power state 14. The format of this field is defined in Figure 113.
2559:2528	O	**Power State 15 Descriptor (PSD15):** This field indicates the characteristics of power state 15. The format of this field is defined in Figure 113.
2591:2560	O	**Power State 16 Descriptor (PSD16):** This field indicates the characteristics of power state 16. The format of this field is defined in Figure 113.
2623:2592	O	**Power State 17 Descriptor (PSD17):** This field indicates the characteristics of power state 17. The format of this field is defined in Figure 113.
2655:2624	O	**Power State 18 Descriptor (PSD18):** This field indicates the characteristics of power state 18. The format of this field is defined in Figure 113.
2687:2656	O	**Power State 19 Descriptor (PSD19):** This field indicates the characteristics of power state 19. The format of this field is defined in Figure 113.
2719:2688	O	**Power State 20 Descriptor (PSD20):** This field indicates the characteristics of power state 20. The format of this field is defined in Figure 113.
2751:2720	O	**Power State 21 Descriptor (PSD21):** This field indicates the characteristics of power state 21. The format of this field is defined in Figure 113.
2783:2752	O	**Power State 22 Descriptor (PSD22):** This field indicates the characteristics of power state 22. The format of this field is defined in Figure 113.
2815:2784	O	**Power State 23 Descriptor (PSD23):** This field indicates the characteristics of power state 23. The format of this field is defined in Figure 113.
2847:2816	O	**Power State 24 Descriptor (PSD24):** This field indicates the characteristics of power state 24. The format of this field is defined in Figure 113.
2879:2848	O	**Power State 25 Descriptor (PSD25):** This field indicates the characteristics of power state 25. The format of this field is defined in Figure 113.
2911:2880	O	**Power State 26 Descriptor (PSD26):** This field indicates the characteristics of power state 26. The format of this field is defined in Figure 113.
2943:2912	O	**Power State 27 Descriptor (PSD27):** This field indicates the characteristics of power state 27. The format of this field is defined in Figure 113.

Bytes	O/M	Description
2975:2944	O	**Power State 28 Descriptor (PSD28):** This field indicates the characteristics of power state 28. The format of this field is defined in Figure 113.
3007:2976	O	**Power State 29 Descriptor (PSD29):** This field indicates the characteristics of power state 29. The format of this field is defined in Figure 113.
3039:3008	O	**Power State 30 Descriptor (PSD30):** This field indicates the characteristics of power state 30. The format of this field is defined in Figure 113.
3071:3040	O	**Power State 31 Descriptor (PSD31):** This field indicates the characteristics of power state 31. The format of this field is defined in Figure 113.
Vendor Specific		
4095:3072	O	Vendor Specific.

Figure 110: Identify – Primary Controller Capabilities Structure

Bytes	Description
1:0	**Controller Identifier (CNTLID):** This field indicates the Controller Identifier of the primary controller.
3:2	**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 is the same as the Port Number field in Link Capabilities Register in the PCI Express Capability structure (refer to section 2.5.6).
4	**Controller Resource Types (CRT):** This field indicates the controller resources types supported. If a controller resource type is supported, then it shall be supported for the primary controller and all associated secondary controllers. 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 8.5.2. 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 8.5.1.
31:5	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 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.
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 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.

Bytes	Description
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

Figure 111 defines a Secondary Controller List. 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 0h or NumVFs specifies a value that does not enable the associated secondary controller).

Figure 111: Secondary Controller List

Bytes	Description
0	**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 0 indicates there are no entries in the list.
31:1	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 112: Secondary Controller Entry

Bytes	Description
1:0	**Secondary Controller Identifier (SCID):** This field indicates the Controller Identifier of the secondary controller described by this entry.
3:2	**Primary Controller Identifier (PCID): T**his field indicates the Controller Identifier of the associated primary controller.
4	**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.
7:5	Reserved
9:8	**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 in the PF's SR-IOV Capabilities structure. If the secondary controller is not an SR-IOV VF, then this field is cleared to zero.
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.
13:12	**Number of VI Flexible Resources Assigned (NVI):** This field indicates the number of VQ Flexible Resources currently assigned to the indicated secondary controller.
31:14	Reserved

Figure 113 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 8.4 on power management.

Figure 113: 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 8.4.3 for more details on each of the defined workloads. This field shall not be “No Workload” unless ACTP is 0000h.</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 0000h 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 (i.e., there are no pending commands, register accesses, background processes, 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 0000h indicates Idle Power is not reported.</td></tr>
<tr><td>127:125</td><td>Reserved</td></tr>
<tr><td>124:120</td><td>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.</td></tr>
<tr><td>119:117</td><td>Reserved</td></tr>
<tr><td>116:112</td><td>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.</td></tr>
<tr><td>111:109</td><td>Reserved</td></tr>
<tr><td>108:104</td><td>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.</td></tr>
<tr><td>103:101</td><td>Reserved</td></tr>
</table>

Bits	Description
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 field indicates whether the controller processes I/O commands in this power state. If this field is cleared to '0', then the controller processes I/O commands in this power state. If this field is set to '1', then the controller does not process I/O commands in this power state. Refer to section 8.4.1.
24	**Max Power Scale (MXPS):** This field indicates the scale for the Maximum Power field. If this field is cleared to '0', then the scale of the Maximum Power field is in 0.01 Watts. If this field 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 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 field. A value of 0h indicates Maximum Power is not reported.

Figure 114 shows the Identify Namespace data structure for the NVM Command Set.

Figure 114: Identify – Identify Namespace Data Structure, NVM Command Set Specific

Bytes	O/M	Description
7:0	M	**Namespace Size (NSZE):** This field indicates the total size of the namespace in logical blocks. A namespace of size *n* consists of LBA 0 through (*n* - 1). The number of logical blocks is based on the formatted LBA size. This field is undefined prior to the namespace being formatted.
15:8	M	**Namespace Capacity (NCAP):** This field indicates the maximum number of logical blocks that may be allocated in the namespace at any point in time. The number of logical blocks is based on the formatted LBA size. This field is undefined prior to the namespace being formatted. This field is used in the case of thin provisioning and reports a value that is smaller than or equal to the Namespace Size. Spare LBAs are not reported as part of this field. A logical block is allocated when it is written with a Write or Write Uncorrectable command. A logical block may be deallocated using the Dataset Management, Sanitize, or Write Zeroes command.
23:16	M	**Namespace Utilization (NUSE):** This field indicates the current number of logical blocks allocated in the namespace. This field is smaller than or equal to the Namespace Capacity. The number of logical blocks is based on the formatted LBA size. When using the NVM command set: A logical block is allocated when it is written with a Write or Write Uncorrectable command. A logical block may be deallocated using the Dataset Management, Sanitize, or Write Zeroes command. A controller may report NUSE equal to NCAP at all times if the product is not targeted for thin provisioning environments.
24	M	**Namespace Features (NSFEAT):** This field defines features of the namespace. Bits 7:4 are reserved. Bit 3 if set to '1' indicates that the non-zero NGUID and non-zero EUI64 fields for this namespace are never reused by the controller. If cleared to '0', then the NGUID and EUI64 values 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 7.11. Bit 2 if set to '1' indicates that the controller supports the Deallocated or Unwritten Logical Block error for this namespace. If cleared to '0', then the controller does not support the Deallocated or Unwritten Logical Block error for this namespace. Refer to section 6.7.1.1. Bit 1 if set to '1' indicates that the fields NAWUN, NAWUPF, and NACWU are defined for this namespace and should be used by the host for this namespace instead of the AWUN, AWUPF, and ACWU fields in the Identify Controller data structure. If cleared to '0', then the controller does not support the fields NAWUN, NAWUPF, and NACWU for this namespace. In this case, the host should use the AWUN, AWUPF, and ACWU fields defined in the Identify Controller data structure in Figure 109. Refer to section 6.4. Bit 0 if set to '1' indicates that the namespace supports thin provisioning. Specifically, the Namespace Capacity reported may be less than the Namespace Size. When this feature is supported and the Dataset Management command is supported then deallocating LBAs shall be reflected in the Namespace Utilization field. Bit 0 if cleared to '0' indicates that thin provisioning is not supported and the Namespace Size and Namespace Capacity fields report the same value.

Bytes	O/M	Description
25	M	**Number of LBA Formats (NLBAF):** This field defines the number of supported LBA data size and metadata size combinations supported by the namespace. LBA formats shall be allocated in order (starting with 0) and packed sequentially. This is a 0's based value. The maximum number of LBA formats that may be indicated as supported is 16. The supported LBA formats are indicated in bytes 128 – 191 in this data structure. The LBA Format fields with an index beyond the value set in this field are invalid and not supported. LBA Formats that are valid, but not currently available may be indicated by setting the LBA Data Size for that LBA Format to 0h. The metadata may be either transferred as part of the LBA (creating an extended LBA which is a larger LBA size that is exposed to the application) or it may be transferred as a separate contiguous buffer of data. The metadata shall not be split between the LBA and a separate metadata buffer. It is recommended that software and controllers transition to an LBA size that is 4KB or larger for ECC efficiency at the controller. If providing metadata, it is recommended that at least 8 bytes are provided per logical block to enable use with end-to-end data protection, refer to section 8.2.
26	M	**Formatted LBA Size (FLBAS):** This field indicates the LBA data size & metadata size combination that the namespace has been formatted with (refer to section 5.23). Bits 7:5 are reserved. Bit 4 if set to '1' indicates that the metadata is transferred at the end of the data LBA, creating an extended data LBA. Bit 4 if cleared to '0' indicates that all of the metadata for a command is transferred as a separate contiguous buffer of data. Bit 4 is not applicable when there is no metadata. Bits 3:0 indicates one of the 16 supported LBA Formats indicated in this data structure.
27	M	**Metadata Capabilities (MC):** This field indicates the capabilities for metadata. Bits 7:2 are reserved. Bit 1 if set to '1' indicates the namespace supports the metadata being transferred as part of a separate buffer that is specified in the Metadata Pointer. Bit 1 if cleared to '0' indicates that the namespace does not support the metadata being transferred as part of a separate buffer. Bit 0 if set to '1' indicates that the namespace supports the metadata being transferred as part of an extended data LBA. Bit 0 if cleared to '0' indicates that the namespace does not support the metadata being transferred as part of an extended data LBA.

<table>
<tr><th>Bytes</th><th>O/M</th><th>Description</th></tr>
<tr><td>28</td><td>M</td><td>End-to-end Data Protection Capabilities (DPC): This field indicates the capabilities for the end-to-end data protection feature. Multiple bits may be set in this field. Refer to section 8.3.

Bits 7:5 are reserved.

Bit 4 if set to ‘1’ indicates that the namespace supports protection information transferred as the last eight bytes of metadata. Bit 4 if cleared to ‘0’ indicates that the namespace does not support protection information transferred as the last eight bytes of metadata.

Bit 3 if set to ‘1’ indicates that the namespace supports protection information transferred as the first eight bytes of metadata. Bit 3 if cleared to ‘0’ indicates that the namespace does not support protection information transferred as the first eight bytes of metadata.

Bit 2 if set to ‘1’ indicates that the namespace supports Protection Information Type 3. Bit 2 if cleared to ‘0’ indicates that the namespace does not support Protection Information Type 3.

Bit 1 if set to ‘1’ indicates that the namespace supports Protection Information Type 2. Bit 1 if cleared to ‘0’ indicates that the namespace does not support Protection Information Type 2.

Bit 0 if set to ‘1’ indicates that the namespace supports Protection Information Type 1. Bit 0 if cleared to ‘0’ indicates that the namespace does not support Protection Information Type 1.</td></tr>
<tr><td>29</td><td>M</td><td>End-to-end Data Protection Type Settings (DPS): This field indicates the Type settings for the end-to-end data protection feature. Refer to section 8.3.

Bits 7:4 are reserved.

Bit 3 if set to ‘1’ indicates that the protection information, if enabled, is transferred as the first eight bytes of metadata. Bit 3 if cleared to ‘0’ indicates that the protection information, if enabled, is transferred as the last eight bytes of metadata.

Bits 2:0 indicate whether Protection Information is enabled and the type of Protection Information enabled. The values for this field have the following meanings:

<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>Protection information is not enabled</td></tr>
<tr><td>001b</td><td>Protection information is enabled, Type 1</td></tr>
<tr><td>010b</td><td>Protection information is enabled, Type 2</td></tr>
<tr><td>011b</td><td>Protection information is enabled, Type 3</td></tr>
<tr><td>100b – 111b</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>30</td><td>O</td><td>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 may only be attached to one controller at a time.</td></tr>
</table>

Bytes	O/M	Description
31	O	**Reservation Capabilities (RESCAP):** This field indicates the reservation capabilities of the namespace. A value of 00h in this field indicates that reservations are not supported by this namespace. Refer to section 8.8 for more details. Bit 7 if set to '1' indicates that Ignore Existing Key is used as defined in revision 1.3 or later of this specification. Bit 7 if cleared to '0' indicates that Ignore Existing Key is used as defined in revision 1.2.1 or earlier of this specification. 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.
32	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). A value of 0 indicates that the namespace is formatted with the format specified by the FLBAS and DPS fields in this data structure and there is no Format NVM command in progress.

<table>
<tr><th>Bytes</th><th>O/M</th><th>Description</th></tr>
<tr><td>33</td><td>O</td><td>Deallocate Logical Block Features (DLFEAT): This field indicates information about features that affect deallocating logical blocks for this namespace.

Bits 7:5 are reserved.

Bit 4 if set to ‘1’ indicates that the Guard field for deallocated logical blocks that contain protection information is set to the CRC for the value read from the deallocated logical block and its metadata (excluding protection information). If cleared to ‘0’ indicates that the Guard field for the deallocated logical blocks that contain protection information is set to FFFFh.

Bit 3 if set to ‘1’ indicates that the controller supports the Deallocate bit in the Write Zeros command for this namespace. If cleared to ‘0’ indicates that the controller does not support the Deallocate bit in the Write Zeros command for this namespace. This bit shall be set to the same value for all namespaces in the NVM subsystem.

Bits 2:0 indicate the values read from a deallocated logical block and its metadata (excluding protection information). The values for this field have the following meanings:

<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>Not reported</td></tr>
<tr><td>001b</td><td>All bytes set to 00h</td></tr>
<tr><td>010b</td><td>All bytes set to FFh</td></tr>
<tr><td>011b – 111b</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>35:34</td><td>O</td><td>Namespace Atomic Write Unit Normal (NAWUN): This field indicates the namespace specific size of the write operation guaranteed to be written atomically to the NVM during normal operation.

A value of 0h indicates that the size for this namespace is the same size as that reported in the AWUN field of the Identify Controller data structure. All other values specify a size in terms of logical blocks using the same encoding as the AWUN field. Refer to section 6.4.</td></tr>
<tr><td>37:36</td><td>O</td><td>Namespace Atomic Write Unit Power Fail (NAWUPF): This field indicates the namespace specific size of the write operation guaranteed to be written atomically to the NVM during a power fail or error condition.

A value of 0h indicates that the size for this namespace is the same size as that reported in the AWUPF field of the Identify Controller data structure. All other values specify a size in terms of logical blocks using the same encoding as the AWUPF field. Refer to section 6.4.</td></tr>
<tr><td>39:38</td><td>O</td><td>Namespace Atomic Compare & Write Unit (NACWU): This field indicates the namespace specific size of the write operation guaranteed to be written atomically to the NVM for a Compare and Write fused command.

A value of 0h indicates that the size for this namespace is the same size as that reported in the ACWU field of the Identify Controller data structure. All other values specify a size in terms of logical blocks using the same encoding as the ACWU field. Refer to section 6.4.</td></tr>
<tr><td>41:40</td><td>O</td><td>Namespace Atomic Boundary Size Normal (NABSN): This field indicates the atomic boundary size for this namespace for the NAWUN value. This field is specified in logical blocks. Writes to this namespace that cross atomic boundaries are not guaranteed to be atomic to the NVM with respect to other read or write commands.

A value of 0h indicates that there are no atomic boundaries for normal write operations. All other values specify a size in terms of logical blocks using the same encoding as the AWUN field. Refer to section 6.4.</td></tr>
</table>

Bytes	O/M	Description
43:42	O	**Namespace Atomic Boundary Offset (NABO):** This field indicates the LBA on this namespace where the first atomic boundary starts. If the NABSN and NABSPF fields are cleared to 0h, then the NABO field shall be cleared to 0h. NABO shall be less than or equal to NABSN and NABSPF. Refer to section 6.4.
45:44	O	**Namespace Atomic Boundary Size Power Fail (NABSPF):** This field indicates the atomic boundary size for this namespace specific to the Namespace Atomic Write Unit Power Fail value. This field is specified in logical blocks. Writes to this namespace that cross atomic boundaries are not guaranteed to be atomic with respect to other read or write commands and there is no guarantee of data returned on subsequent reads of the associated logical blocks. A value of 0h indicates that there are no atomic boundaries for power fail or error conditions. All other values specify a size in terms of logical blocks using the same encoding as the AWUPF field. Refer to section 6.4.
47:46	O	**Namespace Optimal IO Boundary (NOIOB):** This field indicates the optimal IO boundary for this namespace. This field is specified in logical blocks. The host should construct read and write commands that do not cross the IO boundary to achieve optimal performance. A value of 0h indicates that no optimal IO boundary is reported.
63:48	O	**NVM Capacity (NVMCAP):** This field indicates the total size of the NVM allocated to this namespace. The value is in bytes. This field shall be supported if Namespace Management and Namespace Attachment commands are supported. Note: This field may not correspond to the logical block size multiplied by the Namespace Size field. Due to thin provisioning or other settings (e.g., endurance), this field may be larger or smaller than the Namespace Size reported.
103:64		Reserved
119:104	O	**Namespace Globally Unique Identifier (NGUID):** This field contains a 128-bit value that is globally unique and assigned to the namespace when the namespace is created. This field remains fixed throughout the life of the namespace and is preserved across namespace and controller operations (e.g., controller reset, namespace format, etc.). This field uses the EUI-64 based 16-byte designator format. Bytes 114:112 contain the 24-bit Organizationally Unique Identifier (OUI) value assigned by the IEEE Registration Authority. Bytes 119:115 contain an extension identifer assigned by the corresponding organization. Bytes 111:104 contain the vendor specific extension identifier assigned by the corresponding organization. See the IEEE EUI-64 guidelines for more information. This field is big endian (refer to section 7.10). The controller shall specify a globally unique namespace identifier in this field or the EUI64 field when the namespace is created. If the controller is not able to allocate a globally unique identifier then this field shall be cleared to 0h. Refer to section 7.11.
127:120	O	**IEEE Extended Unique Identifier (EUI64):** This field contains a 64-bit IEEE Extended Unique Identifier (EUI-64) that is globally unique and assigned to the namespace when the namespace is created. This field remains fixed throughout the life of the namespace and is preserved across namespace and controller operations (e.g., controller reset, namespace format, etc.). The EUI-64 is a concatenation of a 24-bit or 36-bit Organizationally Unique Identifier (OUI or OUI-36) value assigned by the IEEE Registration Authority and an extension identifier assigned by the corresponding organization. See the IEEE EUI-64 guidelines for more information. This field is big endian (refer to section 7.10). The controller shall specify a globally unique namespace identifier in this field or the NGUID field when the namespace is created. If the controller is not able to allocate a globally unique 64-bit identifier then this field shall be cleared to 0h. Refer to section 7.11.

Bytes	O/M	Description
131:128	M	**LBA Format 0 Support (LBAF0):** This field indicates the LBA format 0 that is supported by the controller. The LBA format field is defined in Figure 115.
135:132	O	**LBA Format 1 Support (LBAF1):** This field indicates the LBA format 1 that is supported by the controller. The LBA format field is defined in Figure 115.
139:136	O	**LBA Format 2 Support (LBAF2):** This field indicates the LBA format 2 that is supported by the controller. The LBA format field is defined in Figure 115.
143:140	O	**LBA Format 3 Support (LBAF3):** This field indicates the LBA format 3 that is supported by the controller. The LBA format field is defined in Figure 115.
147:144	O	**LBA Format 4 Support (LBAF4):** This field indicates the LBA format 4 that is supported by the controller. The LBA format field is defined in Figure 115.
151:148	O	**LBA Format 5 Support (LBAF5):** This field indicates the LBA format 5 that is supported by the controller. The LBA format field is defined in Figure 115.
155:152	O	**LBA Format 6 Support (LBAF6):** This field indicates the LBA format 6 that is supported by the controller. The LBA format field is defined in Figure 115.
159:156	O	**LBA Format 7 Support (LBAF7):** This field indicates the LBA format 7 that is supported by the controller. The LBA format field is defined in Figure 115.
163:160	O	**LBA Format 8 Support (LBAF8):** This field indicates the LBA format 8 that is supported by the controller. The LBA format field is defined in Figure 115.
167:164	O	**LBA Format 9 Support (LBAF9):** This field indicates the LBA format 9 that is supported by the controller. The LBA format field is defined in Figure 115.
171:168	O	**LBA Format 10 Support (LBAF10):** This field indicates the LBA format 10 that is supported by the controller. The LBA format field is defined in Figure 115.
175:172	O	**LBA Format 11 Support (LBAF11):** This field indicates the LBA format 11 that is supported by the controller. The LBA format field is defined in Figure 115.
179:176	O	**LBA Format 12 Support (LBAF12):** This field indicates the LBA format 12 that is supported by the controller. The LBA format field is defined in Figure 115.
183:180	O	**LBA Format 13 Support (LBAF13):** This field indicates the LBA format 13 that is supported by the controller. The LBA format field is defined in Figure 115.
187:184	O	**LBA Format 14 Support (LBAF14):** This field indicates the LBA format 14 that is supported by the controller. The LBA format field is defined in Figure 115.
191:188	O	**LBA Format 15 Support (LBAF15):** This field indicates the LBA format 15 that is supported by the controller. The LBA format field is defined in Figure 115.
383:192		Reserved
4095:384	O	Vendor Specific

Figure 115: Identify – LBA Format Data Structure, NVM Command Set Specific

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:26</td><td>Reserved</td></tr>
<tr><td>25:24</td><td>Relative Performance (RP): This field indicates the relative performance of the LBA format indicated relative to other LBA formats supported by the controller. Depending on the size of the LBA and associated metadata, there may be performance implications. The performance analysis is based on better performance on a queue depth 32 with 4KB read workload. The meanings of the values indicated are included in the following table.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>Best performance</td></tr>
<tr><td>01b</td><td>Better performance</td></tr>
<tr><td>10b</td><td>Good performance</td></tr>
<tr><td>11b</td><td>Degraded performance</td></tr>
</table></td></tr>
<tr><td>23:16</td><td>LBA Data Size (LBADS): This field indicates the LBA data size supported. The value is reported in terms of a power of two (2^n). A value smaller than 9 (i.e. 512 bytes) is not supported. If the value reported is 0h then the LBA format is not supported / used or is not currently available.</td></tr>
<tr><td>15:00</td><td>Metadata Size (MS): This field indicates the number of metadata bytes provided per LBA based on the LBA Data Size indicated. If there is no metadata supported, then this field shall be cleared to 00h.

If metadata is supported, then the namespace may support the metadata being transferred as part of an extended data LBA or as part of a separate contiguous buffer. If end-to-end data protection is enabled, then the first eight bytes or last eight bytes of the metadata is the protection information.</td></tr>
</table>

Figure 116: Identify – Namespace Identification Descriptor

<table>
<tr><th>Byte</th><th>Description</th></tr>
<tr><td>00h</td><td>Namespace Identifier Type (NIDT): This field indicates the data type contained in the Namespace Identifier field as defined in the following table.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0h</td><td>Reserved</td></tr>
<tr><td>1h</td><td>IEEE Extended Unique Identifier: The NID field contains a copy of the EUI64 field in the Identify Namespace structure (refer to Figure 114). If the EUI64 field of the Identify Namespace structure is not supported, the controller shall not report a value of type 1h.

For a Namespace Identifier Descriptor of type 1h the Namespace Identifier Length (NIDL) shall be set to 8h.</td></tr>
<tr><td>2h</td><td>Namespace Globally Unique Identifier: The NID field contains a copy of the NGUID field in the Identify Namespace structure. If the NGUID field of the Identify Namespace structure is not supported, the controller shall not report a value of type 2h.

For a Namespace Identifier Descriptor of type 2h the Namespace Identifier Length (NIDL) shall be set to 10h.</td></tr>
<tr><td>3h</td><td>Namespace UUID: The NID field contains a 128-bit Universally Unique Identifier (UUID) as specified in RFC4122.

For a Namespace Identifier Descriptor of type 3h the Namespace Identifier Length (NIDL) shall be set to 10h.</td></tr>
<tr><td>4h - FFh</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>01h</td><td>Namespace Identifier Length (NIDL): This field contains the length in bytes of the Namespace Identifier field below. The total length of the Namespace Identifier Descriptor in bytes is the value in this field plus four. If this field is set to 0h it indicates the end of the Namespace Identifier Descriptor list.</td></tr>
<tr><td>02h – 03h</td><td>Reserved</td></tr>
<tr><td>04h – (NIDL + 03h)</td><td>Namespace Identifier (NID): This field contains a value that is globally unique and assigned to the namespace when the namespace is created. This field remains fixed throughout the life of the namespace and is preserved across namespace and controller operations (e.g., controller reset, namespace format, etc.). The exact 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>

5.15.1 Command Completion

A completion queue entry is posted to the Admin Completion Queue if the Identify data structure has been transferred to the memory buffer indicated in PRP Entry 1.

5.16 Keep Alive command

The Keep Alive command and associated functionality is used by the host to determine that the controller is operational and 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 execute commands.

If a Keep Alive Timeout has been enabled on the Admin Queue, the Keep Alive Timer is reset when this command is executed.

All command specific fields are reserved.

5.16.1 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.

5.17 NVMe-MI Receive command

The NVMe-MI Receive command transfers an NVMe-MI Response Message from the controller to the host that corresponds to an NVMe-MI Request Message that was previously submitted to the controller (refer to section 5.18). Refer to the NVM Express Management Interface (NVMe-MI) specification for the format and servicing of the NVMe-MI Response Message.

The fields used are Data Pointer and Command Dword 10 fields. All other command specific fields are reserved.

Figure 117: NVMe-MI Receive – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the the start of the data buffer that contains an NVMe-MI Response Message. Refer to Figure 11 for the definition of this field.

Figure 118: NVMe-MI Receive – Command Dword 10

Bit	Description
31:08	Reserved
07:00	**NVMe-MI Specific 0 (NMSP0):** The value of this field contains bits 07:00 of the NVMe-MI Specific field as defined in the NVMe-MI specification.

5.17.1 Command Completion

When the command is completed successfully, the controller shall post a completion queue entry to the Admin Completion Queue indicating successful completion status. An NVMe-MI Receive command is completed successfully when the NVMe-MI Response Message has been transferred to the data buffer pointed to by Data Pointer.

The definition of Dword 0 of the completion queue entry is in Figure 119.

Figure 119: NVMe-MI Receive – Completion Queue Entry Dword 0

Bit	Description
31:00	**Number of Dwords (NUMD):** This field specifies the length of the data structure returned in Dwords. This is a 0's based value.

5.18 NVMe-MI Send command

The NVMe-MI Send command is used to transfer an NVMe-MI Request Message to the controller. Refer to the NVM Express Management Interface (NVMe-MI) specification for the format and servicing of the NVMe-MI Request Message.

The fields used are Data Pointer, Command Dword 10, and Command Dword 11 fields. All other command specific fields are reserved.

Figure 120: NVMe-MI Send – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer that contains an NVMe-MI Request Message. Refer to Figure 11 for the definition of this field.

Figure 121: NVMe-MI Send – Command Dword 10

Bit	Description
31:08	Reserved
07:00	**NVMe-MI Specific 0 (NMSP0):** The value of this field contains bits 07:00 of the NVMe-MI Specific field as defined in the NVMe-MI specification.

Figure 122: NVMe-MI Send – Command Dword 11

Bit	Description
31:00	**Number of Dwords (NUMD):** This field specifies the length of the data structure transferred in Dwords. This is a 0's based value.

5.18.1 Command Completion

When the command is completed successfully, the controller shall post a completion queue entry to the Admin Completion Queue indicating successful completion status. An NVMe-MI Send command is completed successfully when the NVMe-MI Request Message has been transferred to the controller.

5.19 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.

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 4096 bytes in size. The data structure used for Controller Attach and Controller Detach is a Controller List (refer to section 4.9). The controllers that are to be attached or detached, respectively, are described in the data structure.

Figure 123: Namespace Attachment – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 11 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 124: Namespace Attachment – Command Dword 10

Bit	Description
31:04	Reserved
03:00	**Select (SEL):** This field selects the type of attachment to perform. **Value** / **Description** 0h / Controller Attach 1h / Controller Detach 2h - Fh / Reserved

5.19.1 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 125. 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 125: 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.

5.20 Namespace Management command

The Namespace Management command is used to manage namespaces, including create and delete operations. Note: The controller continues to execute commands submitted to I/O Submission Queues while this operation is in progress.

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 any controller as it 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 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 (if Namespace Attribute Notices are supported).

The data structure used for the create operation is defined in Figure 129 and has the same format as the Identify Namespace data structure defined in Figure 114. After successful completion of a Namespace Management command with the create operation, the namespace is formatted with the specified attributes. The fields that host software may specify in the create operation are defined in Figure 126. Fields that are reserved shall be cleared to 0h by host software. There is no data structure transferred for the delete operation.

Figure 126: Namespace Management – Host Software Specified Fields

Bytes	Description	Host Specified
7:0	Namespace Size (NSZE)	Yes
15:8	Namespace Capacity (NCAP)	Yes
25:16	Reserved	
26	Formatted LBA Size (FLBAS)	Yes
28:27	Reserved	
29	End-to-end Data Protection Type Settings (DPS)	Yes
30	Namespace Multi-path I/O and Namespace Sharing Capabilities (NMIC)	Yes
383:31	Reserved	

The Namespace Management command uses the Data Pointer and Dword 10 fields. All other command specific fields are reserved.

The Namespace Identifier (CDW1.NSID) field is used as follows for create and delete operations:

- Create: The CDW1.NSID field is reserved for this operation; host software shall set this field to a value of 0h. The controller shall select an available Namespace Identifier to use for the operation.
- Delete: This field specifies the previously created namespace to delete in this operation. Specifying a value of FFFFFFFFh may be used to delete all namespaces in the NVM subsystem.

Figure 127: Namespace Management – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 11 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 128: Namespace Management – Command Dword 10

Bit	Description
31:04	Reserved
03:00	**Select (SEL):** This field selects the type of management operation to perform. **Value** / **Description** 0h – Create 1h – Delete 2h - Fh – Reserved

Figure 129: Namespace Management – Data Structure for Create

Bytes	Description
383:0	**Identify Namespace:** The fields set by host software are specified in Figure 126. Host software shall set reserved fields to 0h.
1023:384	Reserved
4095:1024	Vendor specific

5.20.1 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 are defined in Figure 130.

Figure 130: Namespace Management – Command Specific Status Values

Value	Description
0Ah	**Invalid Format:** The LBA Format specified is not supported. This may be due to various conditions, including: 1) specifying an invalid LBA Format number, or 2) enabling protection information when there is not sufficient metadata per LBA, or 3) the specified format is not available in the current configuration, or 4) invalid security state (refer to TCG SIIS), etc.
15h	**Namespace Insufficient Capacity:** Creating the namespace requires more free space than is currently available. The Command Specific Information field of the Error Information Log specifies the total amount of 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.

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 131.

Figure 131: Namespace Management – Completion Queue Entry Dword 0

Bit	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.

5.21 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, Command Dword 11, Command Dword 12, Command Dword 13, Command Dword 14, and Command Dword 15 fields. All other command specific fields are reserved.

Figure 132: Set Features – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 11 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 133: Set Features – Command Dword 10

Bit	Description
31	**Save (SV):** This field 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 109 whether this field is supported. If the Feature Identifer specified in the Set Features command is not saveable by the controller and the controller recieves a Set Features command with the Save bit set to one, then the command shall be aborted with a status of Feature Identifer Not Saveable.
30:08	Reserved
07:00	**Feature Identifier (FID):** This field indicates the identifier of the Feature that attributes are being specified for.

5.21.1 Feature Specific Information

Figure 134 defines the Features that may be configured with Set Features and retrieved with Get Features. Figure 135 defines Features that are specific to the NVM Command Set. 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. Feature values that are not persistent across power cycles and resets are reset to their default values as part of a controller reset operation. The default value for each Feature is vendor specific and set by the manufacturer unless otherwise specified; it is not changeable. For more information on Features, including default, saveable, and current value definitions, refer to section 7.8.

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 is successfully completed shall utilize the new settings for the associated Feature. To ensure that a Feature applies to all subsequent commands, commands being processed should be completed prior to issuing the Set Features command.

Figure 134: Set Features – Feature Identifiers

Feature Identifier	O/M[6]	Persistent Across Power Cycle and Reset[2]	Uses Memory Buffer for Attributes	Description
00h				Reserved
01h	M	No	No	Arbitration
02h	M	No	No	Power Management
03h	O	Yes	Yes	LBA Range Type
04h	M	No	No	Temperature Threshold
05h	M	No	No	Error Recovery
06h	O	No	No	Volatile Write Cache
07h	M	No	No	Number of Queues
08h	NOTE 5	No	No	Interrupt Coalescing
09h	NOTE 5	No	No	Interrupt Vector Configuration
0Ah	M	No	No	Write Atomicity Normal
0Bh	M	No	No	Asynchronous Event Configuration
0Ch	O	No	Yes	Autonomous Power State Transition
0Dh	O	No[3]	No[4]	Host Memory Buffer
0Eh	O	No	Yes	Timestamp
0Fh	O	No	No	Keep Alive Timer
10h	O	Yes	No	Host Controlled Thermal Management
11h	O	No	No	Non-Operational Power State Config
12h – 77h				Reserved
78h – 7Fh		Refer to the NVMe Management Interface Specification for definition.		
80h – BFh				Command Set Specific (Reserved)
C0h – FFh				Vendor Specific[1]

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 7.8). If the feature is saveable, then this column is not used and any feature may be configured to be saved across power cycles and reset.
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, but it does use a memory buffer for Get Features. Refer to section 8.9.
5. The feature is mandatory for NVMe over PCIe. This feature is not supported for NVMe over Fabrics.
6. O/M: O = Optional, M = Mandatory.

Figure 135: Set Features, NVM Command Set Specific – Feature Identifiers

Feature Identifier	O/M[6]	Persistent Across Power States and Reset[1]	Uses Memory Buffer for Attributes	Description
80h	O	Yes	No	Software Progress Marker
81h	O[2]	No	Yes	Host Identifier
82h	O[3]	No	No	Reservation Notification Mask
83h	O[3]	Yes	No	Reservation Persistance
84h – BFh				Reserved

NOTES:
1. This column is only valid if bit 4 in the Optional NVM Command Support field of the Identify Controller data structure in Figure 109 is cleared to '0'.
2. Mandatory if reservations are supported as indicated in the Identify Controller data structure.
3. 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.
4. O/M: O = Optional, M = Mandatory.

5.21.1.1 Arbitration (Feature Identifier 01h)

This Feature controls command arbitration. Refer to section 4.11 for command arbitration details. The attributes are indicated in Command Dword 11.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 136 are returned in Dword 0 of the completion queue entry for that command.

Figure 136: Arbitration & Command Processing – Command Dword 11

Bit	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 launch at one time from a particular Submission Queue. This value is specified as 2^n. A value of 111b indicates no limit. Thus, the possible settings are 1, 2, 4, 8, 16, 32, 64, or no limit.

5.21.1.2 Power Management (Feature Identifier 02h)

This Feature allows the host to configure the power state. The attributes are indicated in Command Dword 11.

After a successful completion of a Set Features command for this feature, the controller shall be in the Power State specified. 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 specified in Figure 137 are returned in Dword 0 of the completion queue entry for that command.

Figure 137: Power Management – Command Dword 11

Bit	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 8.4.3 for more details.
04:00	**Power State (PS):** This field indicates the new power state into which the controller should 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.

5.21.1.3 LBA Range Type (Feature Identifier 03h), (Optional)

This feature indicates the type and attributes of LBA ranges that are part of the specified namespace. The LBA range information may be used by a driver to determine if it may utilize a particular LBA range; the information is not exposed to higher level software.

This is optional information that is not required for proper behavior of the system. However, it may be utilized to avoid unintended software issues. For example, if the LBA range indicates that it is a RAID volume then a driver that does not have RAID functionality should not utilize that LBA range (including not overwriting the LBA range). The optional information may be utilized by the driver to determine whether the LBA Range should be exposed to higher level software.

The LBA Range Type uses Command Dword 11 and specifies the type and attribute information in the data structure indicated in Figure 139. The data structure is 4096 bytes in size and shall be physically contiguous.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 138 are returned in Dword 0 of the completion queue entry and the LBA Range Type data structure specified in Figure 139 is returned in the data buffer for that command.

Figure 138: LBA Range Type – Command Dword 11

Bit	Description
31:06	Reserved
05:00	**Number of LBA Ranges (NUM):** This field specifies the number of LBA ranges specified in this command. This is a 0's based value. This field is used for the Set Features command only and is ignored for the Get Features command for this Feature.

Each entry in the LBA Range Type data structure is defined in Figure 139. The LBA Range feature is a set of 64 byte entries; the number of entries is indicated as a command parameter, the maximum number of entries is 64. The LBA ranges shall not overlap. If the LBA ranges overlap, the controller should return an

error of Overlapping Range. All unused entries in the LBA Range Type data structure shall be cleared to all zeroes for both Get Features and Set Features.

The default value for this feature should clear the Number of LBA Ranges field to 00h and initialize the LBA Range Type data structure to contain a single entry with:

- Type field cleared to 00h,
- Attributes field set to 01h,
- Starting LBA field cleared to 0h,
- Number of Logical Blocks field set to indicate the number of LBAs in the namespace, and
- GUID field set to a globally unique identifier.

Figure 139: LBA Range Type – Data Structure Entry

<table>
<tr><th>Byte</th><th>Description</th></tr>
<tr><td>00</td><td>Type (Type): Specifies the Type of the LBA range. The Types are listed below.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>00h</td><td>Reserved</td></tr>
<tr><td>01h</td><td>Filesystem</td></tr>
<tr><td>02h</td><td>RAID</td></tr>
<tr><td>03h</td><td>Cache</td></tr>
<tr><td>04h</td><td>Page / swap file</td></tr>
<tr><td>05h – 7Fh</td><td>Reserved</td></tr>
<tr><td>80h - FFh</td><td>Vendor Specific</td></tr>
</table></td></tr>
<tr><td>01</td><td>Attributes: Specifies attributes of the LBA range. Each bit defines an attribute.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>If set to ‘1’, the LBA range may be overwritten. If cleared to ‘0’, the area should not be overwritten.</td></tr>
<tr><td>1</td><td>If set to ‘1’, the LBA range should be hidden from the OS / EFI / BIOS. If cleared to ‘0’, the area should be visible to the OS / EFI / BIOS.</td></tr>
<tr><td>2 – 7</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>15:02</td><td>Reserved</td></tr>
<tr><td>23:16</td><td>Starting LBA (SLBA): This field specifies the 64-bit address of the first logical block that is part of this LBA range.</td></tr>
<tr><td>31:24</td><td>Number of Logical Blocks (NLB): This field specifies the number of logical blocks that are part of this LBA range. This is a 0’s based value.</td></tr>
<tr><td>47:32</td><td>Unique Identifier (GUID): This field is a global unique identifier that uniquely specifies the type of this LBA range. Well known Types may be defined and are published on the NVM Express website.</td></tr>
<tr><td>63:48</td><td>Reserved</td></tr>
</table>

The host storage driver should expose all LBA ranges that are not set to be hidden from the OS / EFI / BIOS in the Attributes field. All LBA ranges that follow a hidden range shall also be hidden; the host storage driver should not expose subsequent LBA ranges that follow a hidden LBA range.

5.21.1.4 Temperature Threshold (Feature Identifier 04h)

A controller may report up to nine temperature values in the SMART / Health Information log (i.e., the Composite Temperature and Temperature Sensor 1 through Temperature Sensor 8). 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 one of the Critical Warning field in the SMART / Health Information Log is set to one. 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 in Figure 109. 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 if WCTEMP is non-zero; otherwise, it 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 140: Temperature Threshold – Command Dword 11

<table>
<tr><th>Bit</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 – 11b</td><td>Reserved</td></tr>
</table></td></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>0000b</td><td>Composite Temperature</td></tr>
<tr><td>0001b</td><td>Temperature Sensor 1</td></tr>
<tr><td>0010b</td><td>Temperature Sensor 2</td></tr>
<tr><td>0011b</td><td>Temperature Sensor 3</td></tr>
<tr><td>0100b</td><td>Temperature Sensor 4</td></tr>
<tr><td>0101b</td><td>Temperature Sensor 5</td></tr>
<tr><td>0110b</td><td>Temperature Sensor 6</td></tr>
<tr><td>0111b</td><td>Temperature Sensor 7</td></tr>
<tr><td>1000b</td><td>Temperature Sensor 8</td></tr>
<tr><td>1001b – 1110b</td><td>Reserved</td></tr>
<tr><td>1111b</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.</td></tr>
</table>

5.21.1.5 Error Recovery (Feature Identifier 05h)

This Feature controls the error recovery attributes. The attributes are indicated in Command Dword 11.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 141 are returned in Dword 0 of the completion queue entry for that command.

Figure 141: Error Recovery – Command Dword 11

Bit	Description
31:17	Reserved
16	**Deallocated or Unwritten Logical Block Error Enable (DULBE):** If set to '1', then the Deallocated or Unwritten Logical Block error is enabled for the namespace specified in CDW1.NSID. If cleared to '0', then the Deallocated or Unwritten Logical Block error is disabled for the namespace specified in CDW1.NSID. Host software shall only enable this error if it is supported for this namespace as indicated in the Namespace Features field of the Identify Namespace data structure. The default value for this field shall be '0'. Refer to section 6.7.1.1.
15:00	**Time Limited Error Recovery (TLER):** Indicates a limited retry timeout value in 100 millisecond units. This applies to I/O commands that support the Limited Retry bit. The timeout starts when error recovery actions have started while processing the command. A value of 0h indicates that there is no timeout. Note: This mechanism is primarily intended for use by host software that may have alternate means of recovering the data.

5.21.1.6 Volatile Write Cache (Feature Identifier 06h), (Optional)

This Feature controls the volatile write cache, if present, on the controller. If a volatile write cache is supported, then this feature shall be supported. The attributes are indicated 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 setting does not apply to that write cache. In that case, this setting has no effect.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 142 are returned in Dword 0 of the completion queue entry for that command.

Figure 142: Volatile Write Cache – Command Dword 11

Bit	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.

5.21.1.7 Number of Queues (Feature Identifier 07h)

This Feature indicates the number of queues that the host requests for this controller. This feature shall only be issued during initialization prior to creation of any I/O Submission and/or Completion Queues. The value allocated shall not change between resets. The attributes are indicated in Command Dword 11.

If a Set Features or Get Features command is submitted for this Feature, the attributes specified in Figure 144 are returned in Dword 0 of the completion queue entry for that command.

Figure 143: Number of Queues – Command Dword 11

Bit	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 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 (indicating 65,535 I/O Completion Queues). If the value specified is 65,535, the controller should return an error 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 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 (indicating 65,535 I/O Submission Queues). If the value specified is 65,535, the controller should return an error 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 144: Number of Queues – Dword 0 of command completion queue entry

Bit	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 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 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.

5.21.1.8 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.

The controller may delay an interrupt if it detects that interrupts are already being processed for this vector. Specifically, if the Completion Queue Head Doorbell register 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 register 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 145 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 145: Interrupt Coalescing – Command Dword 11

Bit	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.

5.21.1.9 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 146 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 this Feature, the host shall configure the specified Interrupt Vector with a valid I/O Completion Queue. If the I/O Completion Queue or Interrupt Vector specified is invalid, the controller should return an error of Invalid Field in Command.

Figure 146: Interrupt Vector Configuration – Command Dword 11

Bit	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.

5.21.1.10 Write Atomicity Normal (Feature Identifier 0Ah)

This Feature controls the operation of the AWUN and NAWUN parameters (refer to section 6.4). The attributes are indicated in Command Dword 11.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 147 are returned in Dword 0 of the completion queue entry for that command.

Figure 147: Write Atomicity Normal – Command Dword 11

Bit	Description
31:01	Reserved
00	**Disable Normal (DN):** If set to '1', then the host specifies that AWUN and NAWUN are not required and that the controller shall only honor AWUPF and NAWUPF. If cleared to '0', then AWUN, NAWUN, AWUPF, and NAWUPF shall be honored by the controller.

5.21.1.11 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 indicated in Command Dword 11.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 148 are returned in Dword 0 of the completion queue entry for that command.

Figure 148: Asynchronous Event Configuration – Command Dword 11

Bit	Description
31:11	Reserved
10	**Telemetry Log Notices:** This field determines whether an asynchronous event notification is sent to the host when the Telemetry Controller-Initiated Data Available field transitions from '0' to '1' 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 field determines whether an asynchronous event notification is sent to the host for a Firmware Activation Starting event (refer to Figure 49). 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 field determines whether an asynchronous event notification is sent to the host for a Namespace Attribute change (refer to Figure 49). 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 93). 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.

5.21.1.12 Autonomous Power State Transition (Feature Identifier 0Ch), (Optional)

This feature configures the settings for autonomous power state transitions, refer to section 8.4.2.

The Autonomous Power State Transition uses Command Dword 11 and specifies the attribute information in the data structure indicated in Figure 149 and the Autonomous Power State Transition data structure consisting of 32 of the entries defined in Figure 150.

If a Get Features command is issued for this Feature, the attributes specified in Figure 149 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 150, is returned in the data buffer for that command.

Figure 149: Autonomous Power State Transition – Command Dword 11

Bit	Description
31:01	Reserved
00	**Autonomous Power State Transition Enable (APSTE):** This field specifies whether autonomous power state transition is enabled. If this field is set to '1', then autonomous power state transitions are enabled. If this field is cleared to '0', then autonomous power state transitions are disabled. This field is cleared to '0' by default.

Each entry in the Autonomous Power State Transition data structure is defined in Figure 150. 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 150: Autonomous Power State Transition – Data Structure Entry

Bit	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 for the controller to autonomously transition to after there is a continuous period of idle time in the current power state that exceeds time specified in the Idle Time Prior to Transition field. The field specified is required to be a non-operational state as described in Figure 113. This field should not specify a power state with higher reported idle power than the current power state.
02:00	Reserved

5.21.1.13 Host Memory Buffer (Feature Identifier 0Dh), (Optional)

This Feature controls the Host Memory Buffer. The attributes are indicated 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 controller to use exclusively. After a successful completion of a Set Features enabling the host memory buffer, the host shall not write to the associated host memory region, buffer size, or descriptor list until the host memory buffer has been disabled.

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. Posting of the completion queue entry for the Set Features command acknowledges that it is safe for the host software to modify the host memory buffer contents. Refer to section 8.9.

Figure 151: Host Memory Buffer – Command Dword 11

Bit	Description
31:02	Reserved
01	**Memory Return (MR):** If set to ‘1’, then the host is returning previously allocated memory the controller used prior to a reset or entering the Runtime D3 state. 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 EHM was cleared to '0'. 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 controller may use the host memory buffer. While cleared to ‘0’, the controller shall not use the host memory buffer.

Figure 152: Host Memory Buffer – Command Dword 12

Bit	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 153: Host Memory Buffer– Command Dword 13

Bit	Description
31:04	**Host Memory Descriptor List Lower Address (HMDLLA):** This field specifies the lower 32 bits of the physical location of the Host Memory Descriptor List for the Host Memory Buffer. This address shall be 16 byte aligned.
03:00	Reserved

Figure 154: Host Memory Buffer – Command Dword 14

Bit	Description
31:00	**Host Memory Descriptor List Upper Address (HMDLUA):** This field specifies the upper 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) points to 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 155. The Host Memory Descriptor List is described in Figure 156.

Figure 155: Host Memory Buffer – Command Dword 15

Bit	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 156: Host Memory Buffer – Host Memory Descriptor List

Bytes	Description
15:0	Host Memory Buffer Descriptor 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

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 157: Host Memory Buffer – Host Memory Buffer Descriptor Entry

Bit	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 lower bits (*n*:0) of this field indicate the offset within the memory page is 0h. If the memory page size is 4KB, then bits 11:00 shall be zero; if the memory page size is 8KB, then bits 12:00 shall be zero, etc.

If a Get Features command is issued for this Feature, the attributes specified in Figure 151 are returned in Dword 0 of the completion queue entry and the Host Memory Buffer Attributes data structure, whose structure is defined in Figure 158, is returned in the data buffer for that command.

Figure 158: Host Memory Buffer – Attributes Data Structure

Byte	Description
3:0	**Host Memory Buffer Size (HSIZE):** This field specifies the size of the host memory buffer allocated in memory page size units.
7:4	**Host Memory Descriptor List Address (HMDLAL):** This field specifies the lower 32 bits of the physical location of the Host Memory Descriptor List for the Host Memory Buffer. This address shall be 16 byte aligned. The lower 4 bits shall be cleared to zero.
11:8	**Host Memory Descriptor List Address (HMDLAU):** This field specifies the upper 32 bits of the physical location of the Host Memory Descriptor List for the Host Memory Buffer. This address shall be 16 byte aligned. The lower 4 bits shall be cleared to zero.
15:12	**Host Memory Descriptor List Entry Count (HMDLEC):** This field specifies the number of valid Host Memory Descriptor Entries.
4095:16	**Reserved**

5.21.1.14 Timestamp (Feature Identifier 0Eh), (Optional)

The Timestamp 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 accuracy of Timestamp values after initialization may be affected by vendor specific factors, such as whether the controller continuously counts after the timestamp is initialized, or whether it stops counting during certain intervals (such as non-operational power states). Timestamp values should not be used for security applications. The use of the Timestamp is beyond the scope of this specification.

If a Set Features command is issued for this Feature, the data structure specified in Figure 159 is transferred in the data buffer for that command, specifying the Timestamp value.

Figure 159: 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 160 is returned in the data buffer for that command.

Figure 160: 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 set 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 1b, 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 ‘0’ 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 – 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>0b</td><td>The controller counted time in milliseconds continuously since the Timestamp value was initialized.</td></tr>
<tr><td>1b</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>

5.21.1.15 Keep Alive Timer (Feature Identifier 0Fh)

This Feature controls the Keep Alive Timer. Refer to section 7.12 for Keep Alive details. The attributes are indicated in Command Dword 11.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 161 are returned in Dword 0 of the completion queue entry for that command.

Figure 161: Keep Alive Timer – Command Dword 11

Bit	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 PCIe and fabrics that do not require use of the Keep Alive feature. For fabrics that require use of the Keep Alive feature, the default value for this field is 1D4C0h (i.e., 120,000 milliseconds or 2 minutes) rounded up to that granularity.

5.21.1.16 Host Controlled Thermal Management (Feature Identifier 10h), (Optional)

This feature configures the settings for the host controlled thermal management feature, refer to section 8.4.5. The host controlled thermal management feature uses Command Dword 11 with the attributes shown in Figure 162.

If a Get Features command is submitted for this feature, then the attributes shown in Figure 162 are returned in Dword 0 of the completion queue entry for that command.

This feature is not namespace specific.

Figure 162: HCTM – Command Dword 11

Bit	Description
31:16	**Thermal Management Temperature 1 (TMT1):** This field specifies the temperature, in degrees Kelvin, 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 zero, 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 109. 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 109, then the command shall fail 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 fail with a status code of Invalid Field in Command.
15:00	**Thermal Management Temperature 2 (TMT2):** This field specifies the temperature, in degrees Kelvin, 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 zero, 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 109. 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 109, then the command shall fail 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 fail with a status code of Invalid Field in Command.

5.21.1.17 Non-Operational Power State Config (Feature Identifier 11h), (Optional)

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 163 are returned in Dword 0 of the completion queue entry for that command.

Figure 163: Non-Operational Power State Config – Command Dword 11

Bit	Description
31:01	Reserved
00	**Non-Operational Power State Permissive Mode Enable (NOPPME):** If NOPPME 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 NOPPME 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 the host attempts to set this field to '1' and the controller does not support Non-Operational Power State Permissive Mode as indicated in the Controller Attributes field of Identify Controller, then the command fails with a status of Invalid Field in Command.

5.21.1.18 Software Progress Marker (Feature Identifier 80h), (Optional) – NVM Command Set Specific

This Feature is a software progress marker. The software progress marker is persistent across power states. For additional details, refer to section 7.6.1.1. 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 indicated in Command Dword 11.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 164 are returned in Dword 0 of the completion queue entry for that command.

Figure 164: Software Progress Marker – Command Dword 11

Bit	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.

5.21.1.19 Host Identifier (Feature Identifier 81h), (Optional[1])

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.

[1] Mandatory if reservations are supported as indicated in the Identify Controller data structure.

The Host Identifier is contained in the data structure indicated in Figure 166. The attributes are specified in Command Dword 11. If a Get Features command is issued for this Feature, the data structure specified in Figure 166 is returned in the data buffer for that command.

Figure 165: Host Identifier – Command Dword 11

Bit	Description
31:01	Reserved
00	**Enable Extended Host Identifier (EXHID):** If set to '1', then the host is using an extended 128-bit Host Identifier. If cleared to '0', then the host is using 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 value of Invalid Field in Command shall be returned. If the NVM subsystem detects that another controller in the NVM subsystem is using a Host Identifier of a different size than specified in this command, a status of Host Identifier Inconsistent Format shall be returned.

Figure 166: Host Identifier – Data Structure Entry

Byte	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 field. 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 zero indicates that the host is not associated with any other controller in the NVM subsystem.

5.21.1.19.1NVMe over PCIe

The Host Identifier is an optional feature in NVMe over PCIe. 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 Set Features 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.

5.21.1.19.2NVMe over Fabrics

The Host Identifier is a mandatory feature in NVMe over Fabrics. 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 of Command Sequence Error. A Get Features command specifying the Host Identifier Feature shall return the value set in the Fabrics Connect command. A Get Features command

specifying a 64-bit Host Identifier (EXHID cleared to '0') shall be aborted with a status of Invalid Field in Command.

5.21.1.20 Reservation Notification Mask (Feature Identifier 82h), (Optional[2])

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 indicated 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 is aborted with status Invalid Field in Command. If a Set Features or Get Features attempts to access the Reservation Notification Mask on a namespace that does not support reservations or is invalid, then the command is aborted with status Invalid Field in Command.

If a Get Features command successfully completes for this Feature, the attributes specified in Figure 167 are returned in Dword 0 of the completion queue entry for that command.

Figure 167: Reservation Notification Configuration – Command Dword 11

Bit	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

5.21.1.21 Reservation Persistence (Feature Identifier 83h), (Optional[3])

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 6.11). The Reservation Persistence feature attributes are indicated 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

[2] 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.

[3] 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.

command that uses a namespace ID of FFFFFFFFh is aborted with status 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 or 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 Invalid Field in Command.

If a Get Features command successfully completes for this Feature Identifier, the attributes specified in Figure 168 are returned in Dword 0 of the completion queue entry for that command

Figure 168: Reservation Persistence Configuration – Command Dword 11

Bit	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.

5.21.2 Command Completion

A completion queue entry is posted to the Admin Completion Queue when the controller has completed setting attributes associated with the Feature. Set Features command specific status values are defined in Figure 169.

Figure 169: 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 is not able to be changed.
0Fh	**Feature Not Namespace Specific:** The Feature Identifier specified is not namespace specific. The Feature Identifier settings apply across all namespaces.
14h	**Overlapping Range:** This error is indicated if the LBA Range Type data structure has overlapping ranges.

5.22 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 8.5 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 does not exist, is a Private Resource, or is currently in use then an error of Invalid Resource Identifier is returned.

Figure 170: Virtualization Management – Command Dword 10

Bit	Description
31:16	**Controller Identifier (CNTLID):** This field indicates the controller for which controller resources are to be modified.

<table>
<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 – 111b</td><td>Reserved</td></tr></table></td></tr>
<tr><td>07:04</td><td>Reserved</td></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. If the Controller Identifier field does not correspond to this primary controller then an error of Invalid Controller Identifier is returned. This value is persistent across power cycles and resets.</td></tr><tr><td>2h – 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 an error 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 an error 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 8.5) or the primary controller is not enabled, then an error of Invalid Secondary Controller State is returned.</td></tr><tr><td>Ah – Fh</td><td>Reserved</td></tr></table></td></tr>
</table>

Figure 171: Virtualization Management – Command Dword 11

Bit	Description
31:16	Reserved
15:00	**Number of Controller Resources (NR):** This field indicates a number of controller resources to allocate or assign.

5.22.1 Command Completion

Command specific status values associated with the Virtualization management command are defined in Figure 172.

Figure 172: 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.
22h	**Invalid Resource Identifier:** At least one of the specified resource identifiers was invalid.

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 173.

Figure 173: Virtualization Management – Completion Queue Entry Dword 0

Bit	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.

5.23 Format NVM command – NVM Command Set Specific

The Format NVM command is used to low level format the NVM media. This is used when the host wants to change the LBA data size and/or metadata size. 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). 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 may request a secure erase of the contents of the NVM. 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 secure erase depend on the attributes that the controller supports for the Format NVM command and the Namespace Identifier specified in the command. The scope for the format operation is defined in Figure 174. The scope for secure erase, if applicable based on the setting of the Secure Erase Settings field in Command Dword 10 is defined in Figure 175.

Figure 174: Format NVM – Format Scope

FNA[1] Bit 0	NSID	Format Operation
0b	FFFFFFFFh	All namespaces attached to the controller
0b	All other valid values	Particular namespace specified
1b	All valid values	All namespaces in the NVM subsystem
NOTES: 1. FNA is the Format NVM Attributes field in the Identify Controller data structure.		

Figure 175: Format NVM – Secure Erase Scope

FNA[1] Bit 1	NSID	Secure Erase
0b	FFFFFFFFh	All namespaces attached to the controller
0b	All other valid values	Particular namespace specified
1b	All valid values	All namespaces in the NVM subsystem
NOTES: 1. FNA is the Format NVM Attributes field in the Identify Controller data structure.		

The Format NVM command shall fail if the controller is in an invalid security state (refer to the appropriate security specification, e.g., TCG SIIS). The Format NVM command may fail if there are outstanding I/O commands to the namespace specified to be formatted. I/O commands for a namespace that has a Format NVM command in progress may fail.

The settings specified in the Format NVM command are reported as part of the Identify Namespace data structure.

The Format NVM command uses the Command Dword 10 field. All other command specific fields are reserved.

Figure 176: Format NVM – Command Dword 10

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>31:12</td><td>Reserved</td></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 LBAs, 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.). The controller may perform a cryptographic erase when a User Data Erase is requested if all user data is encrypted.</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 – 111b</td><td>Reserved</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>08</td><td>Protection Information Location (PIL): If set to ‘1’ and protection information is enabled, then protection information is transferred as the first eight bytes of metadata. If cleared to ‘0’ and protection information is enabled, then protection information is transferred as the last eight bytes of metadata. This setting is reported in the Formatted LBA Size field of the Identify Namespace data structure.</td></tr>
<tr><td>07:05</td><td>Protection Information (PI): This field specifies whether end-to-end data protection is enabled and the type of protection information. The values for this field have the following meanings:

<table><tr><th>Value</th><th>Definition</th></tr><tr><td>000b</td><td>Protection information is not enabled</td></tr><tr><td>001b</td><td>Protection information is enabled, Type 1</td></tr><tr><td>010b</td><td>Protection information is enabled, Type 2</td></tr><tr><td>011b</td><td>Protection information is enabled, Type 3</td></tr><tr><td>100b – 111b</td><td>Reserved</td></tr></table>
When end-to-end data protected is enabled, the host shall specify the appropriate protection information in the Read, Write, or Compare commands.</td></tr>
<tr><td>04</td><td>Metadata Settings (MSET): This field is set to ‘1’ if the metadata is transferred as part of an extended data LBA. This field is cleared to ‘0’ if the metadata is transferred as part of a separate buffer. The metadata may include protection information, based on the Protection Information (PI) field. If the Metadata Size for the LBA Format selected is 0h, then this field is not applicable.</td></tr>
<tr><td>03:00</td><td>LBA Format (LBAF): This field specifies the LBA format to apply to the NVM media. This corresponds to the LBA formats indicated in the Identify command, refer to Figure 114 and Figure 115. Only supported LBA formats shall be selected.</td></tr>
</table>

5.23.1 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 are defined in Figure 177.

Figure 177: Format NVM – Command Specific Status Values

Value	Description
Ah	**Invalid Format:** The format specified is invalid. This may be due to various conditions, including: 1) specifying an invalid LBA Format number, or 2) enabling protection information when there is not sufficient metadata per LBA 3) the specified format is not available in the current configuration, or 4) invalid security state (refer to TCG SIIS), etc.

5.24 Sanitize command – NVM Command Set Specific

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 8.15 for details on the sanitize operation.

When a sanitize operation starts on any controller, all controllers in the NVM subsystem:

- Shall clear any outstanding Sanitize Operation Completed asynchronous event;
- Shall update the Sanitize Status log (refer to section 5.14.1.9.2);
- Shall abort any command (submitted or in progress) not allowed during a sanitize operation with a status of Sanitize In Progress (refer to section 8.15.1);
- Should suspend power management activities; and
- Shall release stream identifiers for any open streams.

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 of Sanitize In Progress (refer to section 8.15.1).

After a sanitize operation fails, all controllers in the NVM subsystem shall abort any command not allowed during a sanitize operation with a status of Sanitize Failed (refer to section 8.15.1) 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 of Sanitize Failed; and
- any Sanitize command issued in unrestricted completion mode shall be aborted with a status of Sanitize Failed.

The Sanitize Capabilities field in the Identify Controller data structure indicates the sanitize operation types supported. If an unsupported sanitize operation type is selected by a Sanitize command then the controller shall abort the command with a status of Invalid Field in Command.

If a firmware activation is pending, then the controller shall abort any Sanitize command with a status of Firmware Activation Requires NVM Subsystem Reset. Activation of new firmware is prohibited during a sanitize operation (refer to section 8.15.1).

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 of Command Not Supported for Queue in CMB.

All sanitize operations (Block Erase, Crypto Erase, 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 of Successful Completion. If the controller completes a Sanitize command with any status 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 178: Sanitize – Command Dword 10

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>31:10</td><td>Reserved</td></tr>
<tr><td>09</td><td>No Deallocate After Sanitize: If set to ‘1’ then the controller shall not deallocate any logical blocks as a result of successfully completing the sanitize operation. If cleared to ‘0’, then the controller should deallocate logical blocks as a result of successfullly completing the sanitize operation. This bit shall be ignored if the Sanitize Action field is set to 001b (i.e., Exit Failure Mode).</td></tr>
<tr><td>08</td><td>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).</td></tr>
<tr><td>07:04</td><td>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 0 specifies 16 overwrite passes. This field shall be ignored unless the Sanitize Action field is set to 011b (i.e., Overwrite).</td></tr>
<tr><td>03</td><td>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).</td></tr>
<tr><td>02:00</td><td>Sanitize Action (SANACT): This field specifies the sanitize action to perform.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>000b</td><td>Reserved</td></tr>
<tr><td>001b</td><td>Exit Failure Mode</td></tr>
<tr><td>010b</td><td>Start a Block Erase sanitize operation</td></tr>
<tr><td>011b</td><td>Start an Overwrite sanitize operation</td></tr>
<tr><td>100b</td><td>Start a Crypto Erase sanitize operation</td></tr>
<tr><td>101b – 111b</td><td>Reserved</td></tr>
</table></td></tr>
</table>

Figure 179: Sanitize – Command Dword 11

Bit	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 8.15.

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 are defined in Figure 180.

Figure 180: Sanitize – Command Specific Status Values

Value	Description
10h	**Firmware Activation Requires NVM Subsystem Reset:** The sanitize operation could not be started because a firmware activation is pending.

5.25 Security Receive command – NVM Command Set Specific

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 SFSC 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 reset occurs.

The fields used are Data Pointer, Command Dword 10, and Command Dword 11 fields. All other command specific fields are reserved.

Figure 181: Security Receive – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 11 for the definition of this field.

Figure 182: Security Receive – Command Dword 10

Bit	Description
31:24	**Security Protocol (SECP):** This field specifies the security protocol as defined in SFSC. The controller shall fail the command with Invalid Parameter indicated 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 SFSC.
15:08	**SP Specific 0 (SPSP0):** The value of this field contains bits 07:00 of the Security Protocol Specific field as defined in SFSC.
07:00	**NVMe Security Specific Field (NSSF):** Refer to Figure 184 for definition of this field for Security Protocol EAh. For all other Security Protocols this field is reserved.

Figure 183: Security Receive – Command Dword 11

Bit	Description
31:00	**Allocation Length (AL):** The value of this field is specific to the Security Protocol as defined in SFSC where INC_512 = 0.

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 set 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 SFSC 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 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 184.

Figure 184: 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 - FFFFh	Reserved	Reserved

5.26 Security Send command – NVM Command Set Specific

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 SFSC.

The fields used are Data Pointer, Command Dword 10, and Command Dword 11 fields. All other command specific fields are reserved.

Figure 185: Security Send – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to Figure 11 for the definition of this field.

Figure 186: Security Send – Command Dword 10

Bit	Description
31:24	**Security Protocol (SECP):** This field specifies the security protocol as defined in SFSC. The controller shall fail the command with Invalid Parameter indicated 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 SFSC.
15:08	**SP Specific 0 (SPSP0):** The value of this field contains bits 07:00 of the Security Protocol Specific field as defined in SFSC.
07:00	**NVMe Security Specific Field (NSSF):** Refer to Figure 184 for definition of this field for Security Protocol EAh. For all other Security Protocols this field is reserved.

Figure 187: Security Send – Command Dword 11

Bit	Description
31:00	**Transfer Length (TL):** The value of this field is specific to the Security Protocol as defined in SFSC where INC_512 = 0.

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.

6 NVM Command Set

An NVM subsystem is comprised of some number of controllers, where each controller may access some number of namespaces, where each namespace is comprised of some number of logical blocks. A logical block is the smallest unit of data that may be read or written from the controller. The logical block data size, reported in bytes, is always a power of two. Logical block sizes may be 512 bytes, 1KB, 2KB, 4KB, 8KB, etc. Supported logical block sizes are reported in the Identify Namespace data structure.

The NVM Command Set includes the commands listed in Figure 188. The following subsections describe the definition for each of these commands. Commands shall only be submitted by the host when the controller is ready as indicated in the Controller Status register (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 NVM commands are defined in section 4.2. The Completion Queue Entry (CQE) structure and the fields that are common to all NVM commands are defined in section 4.6. The command specific fields in the SQE and CQE structures (i.e., SQE Command Dwords 10-15 and CQE Dword 0) for the NVM Command Set are defined in this section.

In the case of Compare, Read, Write, and Write Zeroes commands, the host may indicate whether a time limit should be applied to error recovery for the operation by setting the Limited Retry (LR) field in the command. The time limit is specified in the Error Recovery feature, specified in section 5.21.1.5. If the host does not specify a time limit should be applied, then the controller should apply all error recovery means to complete the operation.

Figure 188: Opcodes for NVM Commands

Opcode by Field			Combined Opcode[2]	O/M[1]	Command[3]
(07)	(06:02)	(01:00)			
Standard Command	Function	Data Transfer[5]			
0b	000 00b	00b	00h	M	Flush
0b	000 00b	01b	01h	M	Write
0b	000 00b	10b	02h	M	Read
0b	000 01b	00b	04h	O	Write Uncorrectable
0b	000 01b	01b	05h	O	Compare
0b	000 10b	00b	08h	O	Write Zeroes
0b	000 10b	01b	09h	O	Dataset Management
0b	000 11b	01b	0Dh	O[4]	Reservation Register
0b	000 11b	10b	0Eh	O[4]	Reservation Report
0b	001 00b	01b	11h	O[4]	Reservation Acquire
0b	001 01b	01b	15h	O[4]	Reservation Release
	Vendor Specific				
1b	na	NOTE 5	80h – FFh	O	Vendor specific

NOTES:
1. O/M definition: O = Optional, M = Mandatory.
2. Opcodes not listed are reserved.
3. All NVM commands use the Namespace Identifier field (CDW1.NSID).
4. Mandatory if reservations are supported as indicated in the Identify Controller data structure.
5. 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.

6.1 Namespaces

A namespace is a collection of logical blocks that range from 0 to the capacity of the namespace – 1. A namespace ID (NSID) is an identifier used by a controller to provide access to a namespace.

Valid NSIDs are the range of possible NSIDs that correspond to a namespace that may exist in the NVM subsystem. Any NSID is valid, except if it is zero or greater than the Number of Namespaces field reported in the Identify Controller data structure. 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 or the broadcast value.

Active NSIDs are valid NSIDs that are attached to the specific controller. Valid NSIDs that are not attached to the specific controller are called inactive. An active NSID becomes inactive when the associated namespace is detached from the specific controller or is deleted.

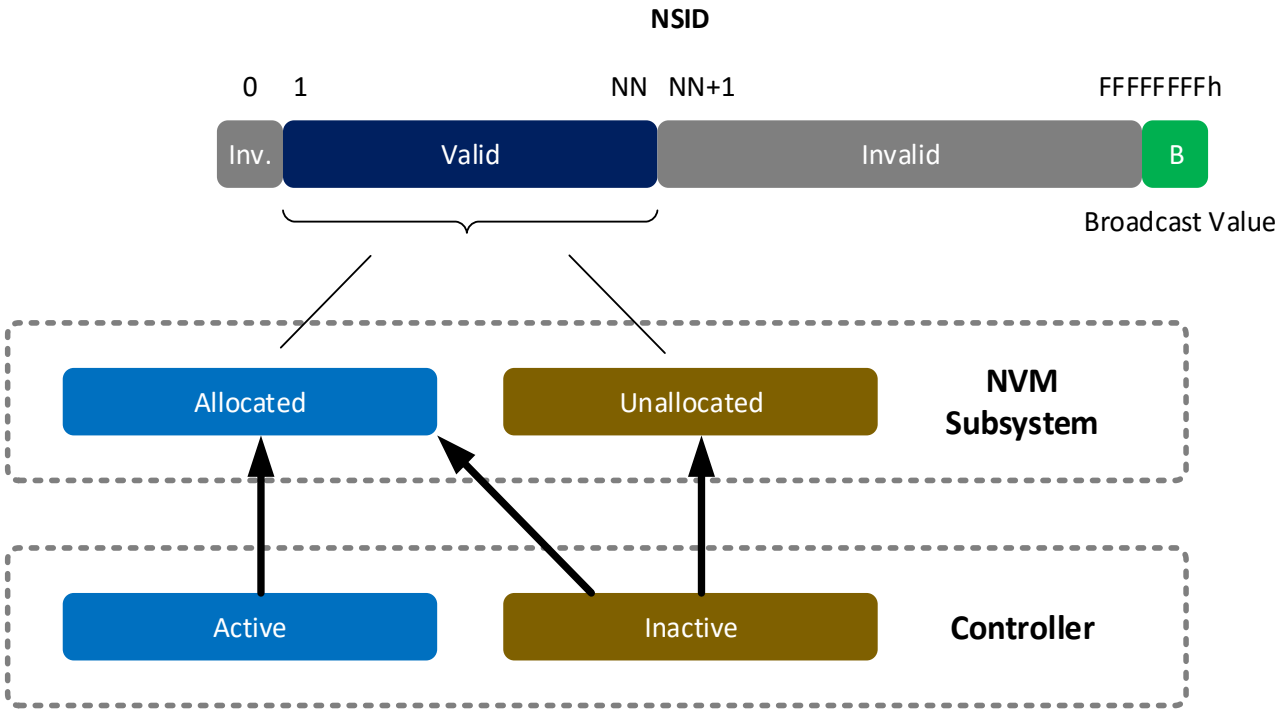
Allocated NSIDs are valid NSIDs that refer to namespaces that currently exist within an NVM subsystem. An allocated NSID may not be attached to any controller. An allocated NSID shall be attached to a controller before host software may submit I/O commands for that namespace on that controller. An allocated NSID becomes unallocated when the associated namespace is deleted.

Unless otherwise noted, specifying an inactive namespace ID in a command that uses the namespace ID shall cause the controller to abort the command with status 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 status Invalid Namespace or Format.

The following table summarizes the valid NSID types Figure 189 visually shows the NSID types and how they relate.

Valid NSID Type	The associated namespace
Active	is attached to this controller
Inactive	is not attached to this controller
Allocated	exists in the NVM subsystem
Unallocated	does not exist in the NVM subsystem

Figure 189: NSID Types



If Namespace Management is supported then Namespace IDs shall be unique within the NVM subsystem (e.g., namespace ID of 3 shall refer to the same physical namespace regardless of the accessing controller). If Namespace Management is not supported then Namespace IDs for private namespaces are not required to be unique.

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

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

1. Issue Identify with the CNS field set to 00h for each valid NSID (based on the Number of Namespaces value in Identify Controller). If a non-zero data structure is returned for a particular NSID, then that is an active NSID.
2. Issue Identify with a CNS field set to 02h to retrieve a list of up to 1024 active NSIDs. If there are more than 1024 active NSIDs, continue to issue Identify with a CNS field set to 02h until all active NSIDs are retrieved.

To determine the allocated NSIDs in the NVM subsystem, the host may Issue Identify with the CNS field set to 10h to retrieve a list of up to 1024 allocated NSIDs. If there are more than 1024 allocated NSIDs, continue to issue Identify with a CNS field set to 10h until all allocated NSIDs are retrieved.

Namespace IDs may change across power off conditions or due to namespace management. However, it is recommended that namespace identifiers remain static in order to avoid issues with EFI or OSes.

The Namespace Size field in the Identify Namespace data structure defines the total size of the namespace in logical blocks (LBA 0 through *n-1*). The Namespace Utilization field in the Identify Namespace data structure defines the number of logical blocks currently allocated in the namespace. The Namespace Capacity field in the Identify data structure defines the maximum number of logical blocks that may be allocated at one time as part of the namespace in a thin provisioning usage model. The following relationship holds: Namespace Size >= Namespace Capacity >= Namespace Utilization.

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 valid namespace from any I/O Submission Queue.

6.2 Fused Operations

The command sequences that may be used in a fused operation for the NVM Command Set are defined in Figure 190. Refer to section 4.10 for details on fused operations.

Figure 190: Supported Fused Operations

Command 1	Command 2	Fused Operation
Compare	Write	Compare and Write

6.2.1 Compare and Write

The Compare and Write fused operation compares the contents of the logical block(s) specified in the Compare command to the data stored at the indicated LBA range. If the compare is successful, then the LBA range is updated with the data provided in the Write command. If the Compare operation is not successful, then the Write operation is aborted with a status of Command Aborted due to Failed Fused Command and the contents in the LBA range are not modified. If the Write operation is not successful, the Compare operation completion status is unaffected.

Note: To ensure the Compare and Write is an atomic operation in a multi-host environment, host software should ensure that the size of a Compare and Write fused operation is no larger than the ACWU/NACWU (refer to section 6.4) and that Atomic Boundaries are respected (refer to section 6.4.3). Controllers may abort a Compare and Write fused operation that is larger than ACWU/NACWU or that crosses an Atomic Boundary with an error of Atomic Write Unit Exceeded.

6.3 Command Ordering Requirements

For all commands which are not part of a fused operation (refer to section 4.10), or for which the write size is greater than AWUN, each command is processed as an independent entity without reference to other commands submitted to the same I/O Submission Queue or to commands submitted to other I/O Submission Queues. Specifically, the controller is not responsible for checking the LBA of a Read or Write command to ensure any type of ordering between commands. For example, if a Read is submitted for LBA *x* and there is a Write also submitted for LBA *x*, there is no guarantee of the order of completion for those commands (the Read may finish first or the Write 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.

The ordering requirements for fused operations are described in section 4.10.

6.4 Atomic Operations

Figure 191 is an overview of the parameters that define the controller's support for atomic operations. These parameters may affect command behavior and execution order based on write size (on a per controller or a per namespace basis).

Figure 191: Atomicity Parameters

	Parameter Name	Value[1]
Controller Atomic Parameters (refer to Identify Controller)	Atomic Write Unit Normal (AWUN)	
	Atomic Write Unit Power Fail (AWUPF)	≤ AWUN
	Atomic Compare and Write Unit (ACWU)	
Namespace Atomic Parameters (refer to Identify Namespace)	Namespace Atomic Write Unit Normal (NAWUN)	≥ AWUN
	Namespace Atomic Write Unit Power Fail (NAWUPF)	≥ AWUPF ≤ NAWUN
	Namespace Atomic Compare and Write Unit (NACWU)	≥ ACWU
Namespace Atomic Boundary Parameters (refer to Identify Namespace)	Namespace Atomic Boundary Size Normal (NABSN)	≥ NAWUN
	Namespace Atomic Boundary Offset (NABO)	≤ NABSN ≤ NABSPF
	Namespace Atomic Boundary Size Power Fail (NABSPF)	≥ NAWUPF

NOTES:
1. When the parameter is supported, the value shall meet the listed condition(s).

The NVM subsystem reports in the Identify Controller data structure the size in logical blocks of the write operation guaranteed to be written atomically under various conditions, including normal operation, power fail, and in a Compare & Write fused operation. The values reported in the Identify Controller data structure are valid across all namespaces with any supported namespace format, forming a baseline value that is guaranteed not to change.

An NVM subsystem may report per namespace values for these fields that are specific to the namespace format in Identify Namespace. If an NVM subsystem reports a per namespace value, it shall be greater than or equal to the corresponding baseline value indicated in Identify Controller.

The values are reported in the fields (Namespace) Atomic Write Unit Normal, (Namespace) Atomic Write Unit Power Fail, and (Namespace) Atomic Compare & Write Unit in Identify Controller or Identify Namespace depending on whether the values are the baseline or namespace specific.

A controller may support Atomic Boundaries that shall not be crossed by an atomic operation. The Namespace Atomic Boundary Parameters (NABSN, NABO, and NABSPF) define these boundaries for a namespace. A namespace supports Atomic Boundaries if NABSN or NABSPF is set to a non-zero value. A namespace that does not support Atomic Boundaries shall clear the NABSN and NABSPF fields to 0h. Namespace Atomicity Parameter and Namespace Atomic Boundary Parameter values may be format specific and may change if the namespace format is modified.

In the case of a shared namespace, operations performed by an individual controller are atomic to the shared namespace at the write atomicity level reported in the corresponding Identify Controller or Identify Namespace data structures of the controller to which the command was submitted.

6.4.1 AWUN/NAWUN

AWUN/NAWUN control the atomicity of command execution in relation to other commands. They impose inter-command serialization of writing of blocks of data to the NVM and prevent blocks of data ending up on the NVM containing partial data from one new command and partial data from one or more other new commands.

If a write command is submitted with size less than or equal to the AWUN/NAWUN value and the write command does not cross an atomic boundary (refer to section 6.4.3), then the host is guaranteed that the write command is atomic to the NVM with respect to other read or write commands. If a write command is submitted with size greater than the AWUN/NAWUN value or crosses an atomic boundary, then there is no guarantee of command atomicity. AWUN/NAWUN does not have any applicability to write errors caused by power failure or other error conditions (refer to Atomic Write Unit Power Fail).

The host may indicate that AWUN and NAWUN are not necessary by configuring the Write Atomicity Normal feature (refer to section 5.21.1.10), which may result in higher performance in some implementations.

6.4.1.1 AWUN/NAWUN Example (Informative)

In this example, AWUN/NAWUN has a value of 2K (equivalent to four 512 byte logical blocks) and the namespace atomic boundary sizes (NABSN and NABSPF) are 0h. The host issues two write commands, each with a length of 2K (i.e., four logical blocks). Command A writes LBAs 0-3 and command B writes LBAs 1-4.

Since the size of both command A and command B is less than or equal to the value of AWUN/NAWUN, the controller serializes these two write commands so that the resulting data in LBAs 0-4 reflects command A followed by command B, or command B followed by command A, but not an intermediate state where some of the logical blocks are written with data from command A and others are written with data from command B. Figure 192 shows valid results of the data in LBAs 0-4 and examples of invalid results (of which there are more possible combinations).

Figure 192: AWUN/NAWUN Example Results

	LBA 0	1	2	3	4	5	6	7
Valid Result	A	A	A	A	B			
Valid Result	A	B	B	B	B			
Invalid Result	A	A	B	B	B			
Invalid Result	A	B	A	A	B			

If the size of write commands A and B is larger than the AWUN/NAWUN value, then there is no guarantee of ordering. After execution of command A and command B, there may be an arbitrary mix of data from command A and command B in the LBA range specified.

6.4.2 AWUPF/NAWUPF

AWUPF and NAWUPF indicate the behavior of the controller if a power fail or other error condition interrupts a write operation causing a torn write. A torn write is a write operation where only some of the logical blocks that are supposed to be written contiguously are actually stored on the NVM, leaving the target logical blocks in an indeterminate state in which some logical blocks contain original data and some logical blocks contain new data from the write operation.

If a write command is submitted with size less than or equal to the AWUPF/NAWUPF value and the write command does not cross an atomic boundary (refer to section 6.4.3), the controller guarantees that if the command fails due to a power failure or other error condition, then subsequent read commands for the logical blocks associated with the write command shall return one of the following:

- All old data (i.e. original data on the NVM in the LBA range addressed by the interrupted write), or
- All new data (i.e. all data to be written to the NVM by the interrupted write)

If a write command is submitted with size greater than the AWUPF/NAWUPF value or crosses an atomic boundary, then there is no guarantee of the data returned on subsequent reads of the associated logical blocks.

6.4.2.1 AWUPF/NAWUPF Example (Informative)

In this example, AWUPF/NAWUPF has a value of 1K (equivalent to two 512 byte logical blocks), AWUN/NAWUN has a value of 2K (equivalent to four 512 byte logical blocks) and the namespace atomic boundary sizes (NABSN and NABSPF) are 0h. Command A writes LBAs 0-1. Figure 193 shows the initial state of the NVM.

Figure 193: AWUPF/NAWUPF Example Initial State of NVM

	LBA 0	1	2	3	4	5	6	7
	C	B	B	B	B			

Command A begins executing but is interrupted by a power failure during the writing of the logical block at LBA 1. Figure 194 describes valid and invalid results.

Figure 194: AWUPF/NAWUPF Example Final State of NVM

	LBA 0	1	2	3	4	5	6	7
Valid Result	A	A	B	B	B			
Valid Result	C	B	B	B	B			
Invalid Result	A	B	B	B	B			
Invalid Result	C	A	B	B	B			
Invalid Result	D	D	B	B	B			

If the size of write command A is larger than the AWUPF/NAWUPF value, then there is no guarantee of the state of the data contained in the specified LBA range after the power fail or error condition.

After a write command has completed, reads for that location which are subsequently submitted shall return the data from that write command and not an older version of the data from previous write commands with the following exception;

If all of the following conditions are met:

a) the controller supports a volatile write cache;
b) the volatile write cache is enabled;
c) the FUA bit for the write is not set;
d) no flush commands, associated with the same namespace as the write, successfully completed before shutdown; and
e) a controller shutdown occurs without completing the normal or abrupt shutdown procedure outlined in section 7.6.2

then subsequent reads for locations written to the volatile write cache that were not written to non-volatile storage may return older data.

6.4.3 Atomic Boundaries

Atomic Boundaries control how the atomicity guarantees defined in section 6.4 are enforced by the controller, with the added constraint of the alignment of the LBA range specified in the command. Atomic Boundaries are defined on a per namespace basis only. The namespace supports Atomic Boundaries if NABSN or NABSPF are set to non-zero values.

To ensure backwards compatibility, the values reported for AWUN, AWUPF, and ACWU shall be set such that they are supported even if a write crosses an atomic boundary. If a controller does not guarantee atomicity across atomic boundaries, the controller shall set AWUN, AWUPF, and ACWU to 0h (1 LBA).

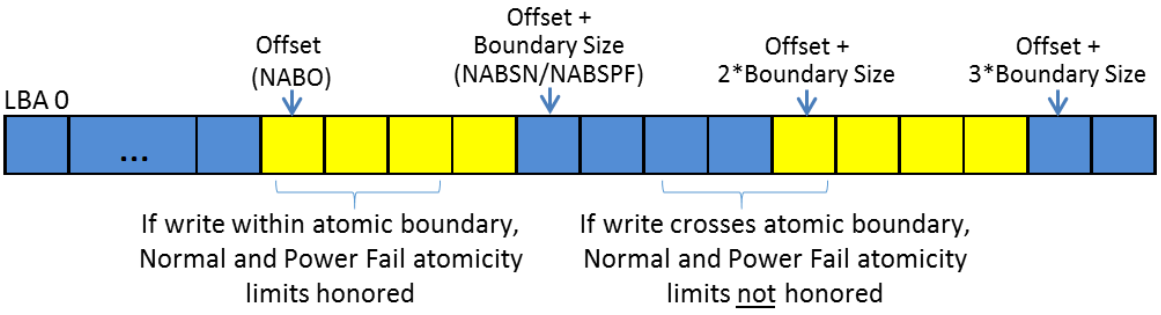
The boundary size shall be greater than or equal to the corresponding atomic write size (i.e., NABSN/NABSPF shall be greater than or equal to NAWUN/NAWUPF, respectively). NABO shall be less than or equal to NABSN and NABSPF.

For Boundary Offset (NABO) and Boundary Size (NABSN or NABSPF), the LBA range in a command is within a Namespace Atomic Boundary if none of the logical block addresses in the range cross: Boundary Offset + (y * Boundary Size); for any integer y ≥ 0.

If a write command crosses the atomic boundary specified by the NABSN value, then the atomicity based on the NAWUN parameters is not guaranteed. If a write command crosses the atomic boundary specified by the NABSPF value, then the atomicity based on the NAWUPF parameters is not guaranteed.

Figure 195 shows an example of the behavior of Atomic Boundaries. Writes to an individual blue or yellow section do not cross an atomic boundary.

Figure 195: Atomic Boundaries Example



6.5 End-to-end Protection Information

The commands that include data transfer may utilize end-to-end data protection. Within these commands, the protection information action and protection information check field is specified as defined in Figure 196.

Figure 196: Protection Information Field Definition

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>03</td><td>Protection Information Action (PRACT): The protection information action field indicates the action to take for the protection information. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.
<table>
<tr><th>PRACT Value</th><th>Metadata Size</th><th>Description</th></tr>
<tr><td>1b</td><td>8 Bytes</td><td>The protection information is stripped (read) or inserted (write).</td></tr>
<tr><td>1b</td><td>> 8 Bytes</td><td>The protection information is passed (read) or replaces the first or last 8 bytes of the metadata (write).</td></tr>
<tr><td>0b</td><td>any</td><td>The protection information is passed (read and write).</td></tr>
</table></td></tr>
<tr><td>02:00</td><td>Protection Information Check (PRCHK): The protection information check field indicates the fields that need to be checked as part of end-to-end data protection processing. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.
<table>
<tr><th>Bit</th><th>Definition</th></tr>
<tr><td>02</td><td>If set to ‘1’ enables protection information checking of the Guard field. If cleared to ‘0’, the Guard field is not checked.</td></tr>
<tr><td>01</td><td>If set to ‘1’ enables protection information checking of the Application Tag field. If cleared to ‘0’, the Application Tag field is not checked.</td></tr>
<tr><td>00</td><td>If set to ‘1’ enables protection information checking of the Logical Block Reference Tag field. If cleared to ‘0’, the Logical Block Reference Tag field is not checked.</td></tr>
</table></td></tr>
</table>

6.6 Compare command

The Compare command reads the logical blocks specified by the command from the medium and compares the data read to a comparison data buffer transferred as part of the command. If the data read from the controller and the comparison data buffer are equivalent with no miscompares, then the command completes successfully. If there is any miscompare, the command completes with an error of Compare Failure.

If metadata is provided, then a comparison is also performed for the metadata, excluding protection information. Refer to section 8.3.

The command uses Command Dword 10, Command Dword 11, Command Dword 12, Command Dword 14, and Command Dword 15 fields. If the command uses PRPs for the data transfer, then the Metadata Pointer, PRP Entry 1, and PRP Entry 2 fields are used. If the command uses SGLs for the data transfer, then the Metadata SGL Segment Pointer and SGL Entry 1 fields are used. All other command specific fields are reserved.

Figure 197: Compare – Metadata Pointer

Bit	Description
63:00	**Metadata Pointer (MPTR):** This field contains the Metadata Pointer, if applicable. Refer to Figure 11 for the definition of this field.

Figure 198: Compare – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the data to use for the compare. Refer to Figure 11 for the definition of this field.

Figure 199: Compare – Command Dword 10 and Command Dword 11

Bit	Description
63:00	**Starting LBA (SLBA):** This field specifies the 64-bit address of the first logical block to compare against as part of the operation. Command Dword 10 contains bits 31:00; Command Dword 11 contains bits 63:32.

Figure 200: Compare – Command Dword 12

Bit	Description
31	**Limited Retry (LR):** If set to '1', the controller should apply limited retry efforts. If cleared to '0', the controller should apply all available error recovery means to retrieve the data for comparison.
30	**Force Unit Access (FUA):** This field specifies that the data read shall be read from non-volatile media.
29:26	**Protection Information Field (PRINFO):** Specifies the protection information action and check field, as defined in Figure 196. The Protection Information Action (PRACT) field shall be cleared to '0'.
25:16	Reserved
15:00	**Number of Logical Blocks (NLB):** This field specifies the number of logical blocks to be compared. This is a 0's based value.

Figure 201: Compare – Command Dword 14

Bit	Description
31:00	**Expected Initial Logical Block Reference Tag (EILBRT):** This field specifies the Initial Logical Block Reference Tag expected value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.

Figure 202: Compare – Command Dword 15

Bit	Description
31:16	**Expected Logical Block Application Tag Mask (ELBATM):** This field specifies the Application Tag Mask expected value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.
15:00	**Expected Logical Block Application Tag (ELBAT):** This field specifies the Application Tag expected value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.

6.6.1 Command Completion

If the command is completed, then the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command. If there are any miscompares between the data read from the NVM media and the data buffer provided, then the command fails with a status code of Compare Failure.

Compare command specific status values are defined in Figure 203.

Figure 203: Compare – Command Specific Status Values

Value	Description
81h	**Invalid Protection Information:** The Protection Information settings specified in the command are invalid.

6.7 Dataset Management command

The Dataset Management command is used by the host to indicate attributes for ranges of logical blocks. This includes attributes like frequency that data is read or written, access size, and other information that may be used to optimize performance and reliability. This command is advisory; a compliant controller may choose to take no action based on information provided.

The command uses Command Dword 10, and Command Dword 11 fields. If the command uses PRPs for the data transfer, then the 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 204: Dataset Management – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the data to use for the command. Refer to Figure 11 for the definition of this field.

Figure 205: Dataset Management – Command Dword 10

Bit	Description
31:08	Reserved
07:00	**Number of Ranges (NR):** Indicates the number of 16 byte range sets that are specified in the command. This is a 0's based value.

Figure 206: Dataset Management – Command Dword 11

Bit	Description
31:03	Reserved
02	**Attribute – Deallocate (AD):** If set to '1' then the NVM subsystem may deallocate all provided ranges. The data returned for a deallocated range is specified in section 6.7.1.1.
01	**Attribute – Integral Dataset for Write (IDW):** If set to '1' then the dataset should be optimized for write access as an integral unit. The host expects to perform operations on all ranges provided as an integral unit for writes, indicating that if a portion of the dataset is written it is expected that all of the ranges in the dataset are going to be written.
00	**Attribute – Integral Dataset for Read (IDR):** If set to '1' then the dataset should be optimized for read access as an integral unit. The host expects to perform operations on all ranges provided as an integral unit for reads, indicating that if a portion of the dataset is read it is expected that all of the ranges in the dataset are going to be read.

If the Dataset Management command is supported, all combinations of attributes specified in Figure 206 may be set.

The data that the Dataset Management command provides is a list of ranges with context attributes. Each range consists of a starting LBA, a length of logical blocks that the range consists of and the context attributes to be applied to that range. The definition of the Dataset Management command Range field is specified in Figure 207. The maximum case of 256 ranges is shown.

Figure 207: Dataset Management – Range Definition

Range	Byte	Field
Range 0	03:00	Context Attributes
	07:04	Length in logical blocks
	15:08	Starting LBA
Range 1	19:16	Context Attributes
	23:20	Length in logical blocks
	31:24	Starting LBA
…		
Range 255	4083:4080	Context Attributes
	4087:4084	Length in logical blocks
	4095:4088	Starting LBA

6.7.1 Context Attributes

The context attributes specified for each range provides information about how the range is intended to be used by host software. The use of this information is optional and the controller is not required to perform any specific action.

Note: The controller is required to maintain the integrity of data on the NVM media regardless of whether the attributes provided by host software are accurate.

Figure 208: Dataset Management – Context Attributes

<table>
<tr><th>Attribute</th><th>Bits</th><th>Description</th></tr>
<tr><td>Command Access Size</td><td>31:24</td><td>Number of logical blocks expected to be transferred in a single Read or Write command from this dataset. A value of 0h indicates no Command Access Size is provided.</td></tr>
<tr><td>Reserved</td><td>23:11</td><td>Reserved</td></tr>
<tr><td>WP: Write Prepare</td><td>10</td><td>If set to ‘1’ then the provided range is expected to be written in the near future.</td></tr>
<tr><td>SW: Sequential Write Range</td><td>09</td><td>If set to ‘1’ then the dataset should be optimized for sequential write access. The host expects to perform operations on the dataset as a single object for writes.</td></tr>
<tr><td>SR: Sequential Read Range</td><td>08</td><td>If set to ‘1’ then the dataset should be optimized for sequential read access. The host expects to perform operations on the dataset as a single object for reads.</td></tr>
<tr><td>Reserved</td><td>07:06</td><td>Reserved</td></tr>
<tr><td>AL: Access Latency</td><td>05:04</td><td>
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>None. No latency information provided.</td></tr>
<tr><td>01b</td><td>Idle. Longer latency acceptable.</td></tr>
<tr><td>10b</td><td>Normal. Typical latency.</td></tr>
<tr><td>11b</td><td>Low. Smallest possible latency.</td></tr>
</table>
</td></tr>
<tr><td>AF: Access Frequency</td><td>03:00</td><td>
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0000b</td><td>No frequency information provided.</td></tr>
<tr><td>0001b</td><td>Typical number of reads and writes expected for this LBA range.</td></tr>
<tr><td>0010b</td><td>Infrequent writes and infrequent reads to the LBA range indicated.</td></tr>
<tr><td>0011b</td><td>Infrequent writes and frequent reads to the LBA range indicated.</td></tr>
<tr><td>0100b</td><td>Frequent writes and infrequent reads to the LBA range indicated.</td></tr>
<tr><td>0101b</td><td>Frequent writes and frequent reads to the LBA range indicated.</td></tr>
<tr><td>0110b – 1111b</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>

6.7.1.1 Deallocate

A logical block that has been deallocated using the Dataset Management command is no longer deallocated when the logical block is written. Read operations do not affect the deallocation status of a logical block. The value read from a deallocated logical block shall be deterministic; specifically, the value returned by subsequent reads of that logical block shall be the same until a write occurs to that logical block.

The values read from a deallocated logical block and its metadata (excluding protection information) shall be all bytes set to 00h, all bytes set to FFh, or the last data written to the associated logical block and its metadata, except that access is prohibited to all data and metadata values written before the most recent successful sanitize operation, if any. The Deallocate Logical Block Features field in the Identify Namespace data structure may report the values read from a deallocated logical block and its metadata.

The values read from a deallocated or unwritten logical block’s protection information field shall:

- have the Guard field value set to FFFFh or set to the CRC for the value read from the deallocated logical block and its metadata (excluding protection information) (e.g., set to 0000h if the value read is all bytes set to 00h); and
- have the Application Tag field value set to FFFFh and the Reference Tag field value set to FFFFFFFFh (indicating the protection information shall not be checked).

Host software may enable an error to be returned if a deallocated or unwritten logical block is read in the Error Recovery feature. If this error is supported for the namespace and enabled, then a read or compare containing a deallocated or unwritten logical block shall fail with the Unwritten or Deallocated Logical Block status code. Note: Legacy software may not handle an error for this case.

Note: The operation of the Deallocate function is similar to the ATA DATA SET MANAGEMENT with Trim feature described in ACS-2 and SCSI UNMAP command described in SBC-3.

6.7.2 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.

Dataset Management command specific status values are defined in Figure 209.

Figure 209: Dataset Management – Command Specific Status Values

Value	Description
80h	**Conflicting Attributes:** The attributes specified in the command are conflicting.
82h	**Attempted Write to Read Only Range:** The controller may optionally report this status if a Deallocate is attempted for a read only range.

6.8 Flush command

The Flush command shall commit data and metadata associated with the specified namespace(s) to non-volatile media. The flush applies to all commands completed prior to the submission of the Flush command. The controller may also flush additional data and/or metadata from any namespace.

All command specific fields are reserved.

6.8.1 Command Completion

If the command is completed, then the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command.

6.9 Read command

The Read command reads data and metadata, if applicable, from the NVM controller for the LBAs indicated. The command may specify protection information to be checked as part of the read operation.

The command uses Command Dword 10, Command Dword 11, Command Dword 12, Command Dword 13, Command Dword 14, and Command Dword 15 fields. If the command uses PRPs for the data transfer, then the Metadata Pointer, PRP Entry 1, and PRP Entry 2 fields are used. If the command uses SGLs for the data transfer, then the Metadata SGL Segment Pointer and SGL Entry 1 fields are used.

Figure 210: Read – Metadata Pointer

Bit	Description
63:00	**Metadata Pointer (MPTR):** This field contains the Metadata Pointer, if applicable. Refer to Figure 11 for the definition of this field.

Figure 211: Read – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies where data is transferred to. Refer to Figure 11 for the definition of this field.

Figure 212: Read – Command Dword 10 and Command Dword 11

Bit	Description
63:00	**Starting LBA (SLBA):** This field indicates the 64-bit address of the first logical block to be read as part of the operation. Command Dword 10 contains bits 31:00; Command Dword 11 contains bits 63: 32.

Figure 213: Read – Command Dword 12

Bit	Description
31	**Limited Retry (LR):** If set to '1', the controller should apply limited retry efforts. If cleared to '0', the controller should apply all available error recovery means to return the data to the host.
30	**Force Unit Access (FUA):** This field indicates that the data read shall be returned from non-volatile media. There is no implied ordering with other commands.
29:26	**Protection Information Field (PRINFO):** Specifies the protection information action and check field, as defined in Figure 196.
25:16	Reserved
15:00	**Number of Logical Blocks (NLB):** This field indicates the number of logical blocks to be read. This is a 0's based value.

Figure 214: Read – Command Dword 13

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>31:08</td><td>Reserved</td></tr>
<tr><td>07:00</td><td>Dataset Management (DSM): This field indicates attributes for the LBA(s) being read.
<table>
<tr><th>Bits</th><th>Attribute</th><th>Definition</th></tr>
<tr><td>07</td><td>Incompressible</td><td>If set to ‘1’, then data is not compressible for the logical blocks indicated. If cleared to ‘0’, then no information on compression is provided.</td></tr>
<tr><td>06</td><td>Sequential Request</td><td>If set to ‘1’, then this command is part of a sequential read that includes multiple Read commands. If cleared to ‘0’, then no information on sequential access is provided.</td></tr>
<tr><td>05:04</td><td>Access Latency</td><td>
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>None. No latency information provided.</td></tr>
<tr><td>01b</td><td>Idle. Longer latency acceptable.</td></tr>
<tr><td>10b</td><td>Normal. Typical latency.</td></tr>
<tr><td>11b</td><td>Low. Smallest possible latency.</td></tr>
</table>
</td></tr>
<tr><td>03:00</td><td>Access Frequency</td><td>
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0000b</td><td>No frequency information provided.</td></tr>
<tr><td>0001b</td><td>Typical number of reads and writes expected for this LBA range.</td></tr>
<tr><td>0010b</td><td>Infrequent writes and infrequent reads to the LBA range indicated.</td></tr>
<tr><td>0011b</td><td>Infrequent writes and frequent reads to the LBA range indicated.</td></tr>
<tr><td>0100b</td><td>Frequent writes and infrequent reads to the LBA range indicated.</td></tr>
<tr><td>0101b</td><td>Frequent writes and frequent reads to the LBA range indicated.</td></tr>
<tr><td>0110b</td><td>One time read. E.g. command is due to virus scan, backup, file copy, or archive.</td></tr>
<tr><td>0111b</td><td>Speculative read. The command is part of a prefetch operation.</td></tr>
<tr><td>1000b</td><td>The LBA range is going to be overwritten in the near future.</td></tr>
<tr><td>1001b – 1111b</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Figure 215: Read – Command Dword 14

Bit	Description
31:00	**Expected Initial Logical Block Reference Tag (EILBRT):** This field specifies the Initial Logical Block Reference Tag expected value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.

Figure 216: Read – Command Dword 15

Bit	Description
31:16	**Expected Logical Block Application Tag Mask (ELBATM):** This field specifies the Application Tag Mask expected value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.
15:00	**Expected Logical Block Application Tag (ELBAT):** This field specifies the Application Tag expected value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.

6.9.1 Command Completion

When the command is completed with success or failure, the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command.

Read command specific status values are defined in Figure 217.

Figure 217: Read – Command Specific Status Values

Value	Description
80h	**Conflicting Attributes:** The attributes specified in the command are conflicting.
81h	**Invalid Protection Information:** The Protection Information settings specified in the command are invalid.

6.10 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 218: Reservation Acquire – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the the location of a data buffer where data is transferred from. Refer to Figure 11 for the definition of this field.

Figure 219: Reservation Acquire – Command Dword 10

Bit	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 221.
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** 000b – Acquire 001b – Preempt 010b – Preempt and Abort 011b - 111b – Reserved

Figure 220: Reservation Acquire Data Structure

Bytes	O/M	Description
7:0	M	**Current Reservation Key (CRKEY):** The field specifies the current reservation key associated with the host.
15:8	M	**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 221: 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
07h-FFh	Reserved

6.10.1 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.

6.11 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. 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 222: Reservation Register – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the location of a data buffer where data is transferred from. Refer to Figure 11 for the definition of this field.

Figure 223: Reservation Register – Command Dword 10

<table>
<tr><th>Bit</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 state associated with the namespace to be modified as a side effect of processing this command.
<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>Set 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>
<tr><td>02:00</td><td>Reservation Register Action (RREGA): This field specifies the registration action that is performed by the command.
<table>
<tr><th>RREGA Value</th><th>Description</th></tr>
<tr><td>000b</td><td>Register Reservation Key</td></tr>
<tr><td>001b</td><td>Unregister Reservation Key</td></tr>
<tr><td>010b</td><td>Replace Reservation Key</td></tr>
<tr><td>011b - 111b</td><td>Reserved</td></tr>
</table></td></tr>
</table>

Figure 224: Reservation Register Data Structure

Bytes	O/M	Description
7:0	M	**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’.
15:8	M	**New Reservation Key (NRKEY):** If the Reservation Register Action is 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.

6.11.1 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.

6.12 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 225: Reservation Release – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the location of a data buffer where data is transferred from. Refer to Figure 11 for the definition of this field.

Figure 226: Reservation Release – Command Dword 10

Bit	Description
31:16	Reserved
15:08	**Reservation Type (RTYPE):** If the Reservation Release Action is 00b (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 it does not match the controller should return an error of Invalid Field In Command. This field is defined in Figure 221.
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 Release Action (RRELA):** This field specifies the registration action that is performed by the command. **RRELA Value** / **Description** 000b – Release 001b – Clear 010b - 111b – Reserved

Figure 227: 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.

6.12.1 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.

6.13 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 231 if the host has selected a 64-bit Host Identifier and the data structure in Figure 232 if the host has selected a 128-bit Host Identifier (refer to section 5.21.1.19).

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 228: Reservation Report – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the location of a data buffer where data is transferred to. Refer to Figure 11 for the definition of this field.

Figure 229: Reservation Report – Command Dword 10

Bit	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 230: Reservation Report – Command Dword 11

Bit	Description
31:01	Reserved
00	**Extended Data Structure (EDS):** If set to '1' then the controller returns the extended data structure defined in Figure 232. If cleared to '0' then the controller returns the data structure defined in Figure 231.

Figure 231: Reservation Status Data Structure

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>3:0</td><td>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.</td></tr>
<tr><td>4</td><td>Reservation Type (RTYPE): This field indicates whether a reservation is held on the namespace. A value of zero 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 221.</td></tr>
<tr><td>6:5</td><td>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.</td></tr>
<tr><td>8:7</td><td>Reserved</td></tr>
<tr><td>9</td><td>Persist Through Power Loss State (PTPLS): This field indicates the Persist Through Power Loss State associated with the namespace.

<table>
<tr><th>PTPLS Value</th><th>Description</th></tr>
<tr><td>0</td><td>Reservations are released and registrants are cleared on a power on.</td></tr>
<tr><td>1</td><td>Reservations and registrants persist across a power loss.</td></tr>
</table>
</td></tr>
<tr><td>23:10</td><td>Reserved</td></tr>
<tr><td>47:24</td><td>Registered Controller DataStructure 0</td></tr>
<tr><td></td><td>.
.
.</td></tr>
<tr><td>24*n+47:
24*(n+1)</td><td>Registered Controller DataStructure n</td></tr>
</table>

Figure 232: Reservation Status Extended Data Structure

Bytes	Description
23:0	Refer to Figure 231 for definition.
63:24	Reserved
127:64	**Registered Controller Extended DataStructure 0**
	. . .
64*(n+1)+63: 64*(n+1)	**Registered Controller Extended DataStructure n**

Figure 233: Registered Controller Data Structure

Bytes	Description
1:0	**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.
2	**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.
7:3	Reserved
15:8	**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 234: Registered Controller Extended Data Structure

Bytes	Description
1:0	**Controller ID (CNTLID):** Refer to Figure 233 for definition.
2	**Reservation Status (RCSTS):** Refer to Figure 233 for definition.
7:3	Reserved
15:8	**Reservation Key (RKEY):** Refer to Figure 233 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

6.13.1 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.

6.14 Write command

The Write command writes data and metadata, if applicable, to the NVM controller for the logical blocks indicated. The host may also specify protection information to include as part of the operation.

The command uses Command Dword 10, Command Dword 11, Command Dword 12, Command Dword 13, Command Dword 14, and Command Dword 15 fields. If the command uses PRPs for the data transfer, then the Metadata Pointer, PRP Entry 1, and PRP Entry 2 fields are used. If the command uses SGLs for the data transfer, then the Metadata SGL Segment Pointer and SGL Entry 1 fields are used.

Figure 235: Write – Metadata Pointer

Bit	Description
63:00	**Metadata Pointer (MPTR):** This field contains the Metadata Pointer, if applicable. Refer to Figure 11 for the definition of this field.

Figure 236: Write – Data Pointer

Bit	Description
127:00	**Data Pointer (DPTR):** This field specifies the location of a data buffer where data is transferred from. Refer to Figure 11 for the definition of this field.

Figure 237: Write – Command Dword 10 and Command Dword 11

Bit	Description
63:00	**Starting LBA (SLBA):** This field indicates the 64-bit address of the first logical block to be written as part of the operation. Command Dword 10 contains bits 31:00; Command Dword 11 contains bits 63:32.

Figure 238: Write – Command Dword 12

Bit	Description
31	**Limited Retry (LR):** If set to '1', the controller should apply limited retry efforts. If cleared to '0', the controller should apply all available error recovery means to write the data to the NVM.
30	**Force Unit Access (FUA):** This field indicates that the data shall be written to non-volatile media before indicating command completion. There is no implied ordering with other commands.
29:26	**Protection Information Field (PRINFO):** Specifies the protection information action and check field, as defined in Figure 196.
25:24	Reserved
23:20	**Directive Type (DTYPE)**: Specifies the Directive Type associated with the Directive Specific field (refer to section 9.1).
19:16	Reserved
15:00	**Number of Logical Blocks (NLB):** This field indicates the number of logical blocks to be written. This is a 0's based value.

Figure 239: Write – Command Dword 13

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>31:16</td><td>Directive Specific (DSPEC): Specifies the Directive Specific value associated with the Directive Type field (refer to section 9.1).</td></tr>
<tr><td>15:08</td><td>Reserved</td></tr>
<tr><td>07:00</td><td>Dataset Management (DSM): This field indicates attributes for the LBA(s) being written.
<table>
<tr><th>Bits</th><th>Attribute</th><th>Definition</th></tr>
<tr><td>07</td><td>Incompressible</td><td>If set to ‘1’, then data is not compressible for the logical blocks indicated. If cleared to ‘0’, then no information on compression is provided.</td></tr>
<tr><td>06</td><td>Sequential Request</td><td>If set to ‘1’, then this command is part of a sequential write that includes multiple Write commands. If cleared to ‘0’, then no information on sequential access is provided.</td></tr>
<tr><td>05:04</td><td>Access Latency</td><td>
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>None. No latency information provided.</td></tr>
<tr><td>01b</td><td>Idle. Longer latency acceptable.</td></tr>
<tr><td>10b</td><td>Normal. Typical latency.</td></tr>
<tr><td>11b</td><td>Low. Smallest possible latency.</td></tr>
</table>
</td></tr>
<tr><td>03:00</td><td>Access Frequency</td><td>
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0000b</td><td>No frequency information provided.</td></tr>
<tr><td>0001b</td><td>Typical number of reads and writes expected for this LBA range.</td></tr>
<tr><td>0010b</td><td>Infrequent writes and infrequent reads to the LBA range indicated.</td></tr>
<tr><td>0011b</td><td>Infrequent writes and frequent reads to the LBA range indicated.</td></tr>
<tr><td>0100b</td><td>Frequent writes and infrequent reads to the LBA range indicated.</td></tr>
<tr><td>0101b</td><td>Frequent writes and frequent reads to the LBA range indicated.</td></tr>
<tr><td>0110b</td><td>One time write. E.g. command is due to virus scan, backup, file copy, or archive.</td></tr>
<tr><td>0111b – 1111b</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Figure 240: Write – Command Dword 14

Bit	Description
31:00	**Initial Logical Block Reference Tag (ILBRT):** This field specifies the Initial Logical Block Reference Tag value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.

Figure 241: Write – Command Dword 15

Bit	Description
31:16	**Logical Block Application Tag Mask (LBATM):** This field specifies the Application Tag Mask value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.
15:00	**Logical Block Application Tag (LBAT):** This field specifies the Application Tag value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.

6.14.1 Command Completion

When the command is completed with success or failure, the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command.

Write command specific errors are defined in Figure 242.

Figure 242: Write – Command Specific Status Values

Value	Description
80h	**Conflicting Attributes:** The attributes specified in the command are conflicting.
81h	**Invalid Protection Information:** The Protection Information settings specified in the command are invalid.
82h	**Attempted Write to Read Only Range:** The LBA range specified contains read-only blocks.

6.15 Write Uncorrectable command

The Write Uncorrectable command is used to mark a range of logical blocks as invalid. When the specified logical block(s) are read after this operation, a failure is returned with Unrecovered Read Error status. To clear the invalid logical block status, a write operation is performed on those logical blocks.

The fields used are Command Dword 10, Command Dword 11, and Command Dword 12 fields. All other command specific fields are reserved.

Figure 243: Write Uncorrectable – Command Dword 10 and Command Dword 11

Bit	Description
63:00	**Starting LBA (SLBA):** This field specifies the 64-bit address of the first logical block to be marked as invalid as part of the operation. Command Dword 10 contains bits 31:00; Command Dword 11 contains bits 63: 32.

Figure 244: Write Uncorrectable – Command Dword 12

Bit	Description
31:16	Reserved
15:00	**Number of Logical Blocks (NLB):** This field specifies the number of logical blocks to be marked as invalid. This is a 0's based value.

6.15.1 Command Completion

If the command is completed, then the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command.

Figure 245: Write Uncorrectable – Command Specific Status Values

Bit	Description
82h	**Attempted Write to Read Only Range:** The LBA range specified contains read-only blocks.

6.16 Write Zeroes command

The Write Zeroes command is used to set a range of logical blocks to zero. After successful completion of this command, the value returned by subsequent reads of logical blocks in this range shall be all bytes set to 00h until a write occurs to this LBA range. The metadata for this command shall be all bytes set to 00h and the protection information for logical blocks written to the media is updated based on CDW12.PRINFO. If the Protection Information Action field (PRACT) is cleared to '0', the metadata for this command shall be all zeroes. If the Protection Information Action field (PRACT) is set to '1', any non-PI related metadata, if it exists, shall be all bytes set to 00h.

If the Deallocate bit (CDW12.DEAC) is set to '1' in a Write Zeroes command, and the namespace supports setting all bytes to 00h in the values read from a deallocated logical block and its metadata (excluding protection information), then the controller:

- should deallocate all logical blocks in the range specified by that command;
- shall return all bytes cleared to 00h in the values read from the deallocated logical blocks and its metadata (excluding protection information); and
- shall return the protection information in the deallocated logical blocks as specified in section 6.7.1.1.

If the Deallocate bit is cleared to '0' in a Write Zeroes command, and the namespace supports setting all bytes to 00h in the values read from a deallocated logical block and its metadata (excluding protection information), then the controller:

- may deallocate any logical blocks in the range specified by that command;
- shall return all bytes cleared to 00h in the values read from the deallocated logical blocks and its metadata (excluding protection information); and
- shall return the protection information in the deallocated logical blocks based on CDW12.PRINFO in that Write Zeroes command.

If the namespace does not support setting all bytes to 00h in the values read from a deallocated logical block and its metadata (excluding protection information), then the controller shall not deallocate any logical blocks in the range specified by a Write Zeroes command.

The fields used are Command Dword 10, Command Dword 11, Command Dword 12, Command Dword 14, and Command Dword 15 fields.

Figure 246: Write Zeroes – Command Dword 10 and Command Dword 11

Bit	Description
63:00	**Starting LBA (SLBA):** This field indicates the 64-bit address of the first logical block to be written as part of the operation. Command Dword 10 contains bits 31:00; Command Dword 11 contains bits 63:32.

Figure 247: Write Zeroes – Command Dword 12

Bit	Description
31	**Limited Retry (LR):** If set to '1', the controller should apply limited retry efforts. If cleared to '0', the controller should apply all available error recovery means to write the data to the NVM.
30	**Force Unit Access (FUA):** This field indicates that the data shall be written to non-volatile media before indicating command completion. There is no implied ordering with other commands.

29:26	**Protection Information Field (PRINFO):** Specifies the protection information action and check field, as defined in Figure 196. The Protection Information Check field (PRCHK) shall be 000b.
25	**Deallocate (DEAC):** If set to '1', then the controller should deallocate logical blocks and may write all bytes set to 00h. If cleared to '0', then the controller may write all bytes set to 00h or may deallocate logical blocks.
24:16	Reserved
15:00	**Number of Logical Blocks (NLB):** This field indicates the number of logical blocks to be written. This is a 0's based value.

Figure 248: Write Zeroes – Command Dword 14

Bit	Description
31:00	**Initial Logical Block Reference Tag (ILBRT):** This field indicates the Initial Logical Block Reference Tag value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.

Figure 249: Write Zeroes – Command Dword 15

Bit	Description
31:16	**Logical Block Application Tag Mask (LBATM):** This field indicates the Application Tag Mask value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.
15:00	**Logical Block Application Tag (LBAT):** This field indicates the Application Tag value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.

6.16.1 Command Completion

When the command is completed with success or failure, the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command.

Write Zeroes command specific status values are defined in Figure 250.

Figure 250: Write Zeroes – Command Specific Status Values

Value	Description
81h	**Invalid Protection Information:** The Protection Information settings specified in the command are invalid.
82h	**Attempted Write to Read Only Range:** The LBA range specified contains read-only blocks.

7 Controller Architecture

7.1 Introduction

Host software submits commands to the controller through pre-allocated Submission Queues. The 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.

The controller fetches the commands from the Submission Queue(s) and transmits them to the NVM subsystem for processing. Except for fused operations, there are no ordering restrictions for processing of the commands within or across Submission Queues. Host software should not place commands in the list that may not be re-ordered arbitrarily. Data may or may not be committed to the NVM media in the order that commands are received.

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 it is issued through. The controller arbitrates across the Submission Queues based on fairness and priority according to the arbitration scheme specified in section 4.11.

Upon completion of the commands by the NVM subsystem, the controller presents completion queue entries to the host through the appropriate Completion Queues. If MSI-X or multiple message MSI is in use, then the interrupt vector indicates the Completion Queue(s) with possible new command completions for the host to process. If pin-based interrupts or single message MSI interrupts are used, host software interrogates the Completion Queue(s) for new completion queue entries. The host updates the CQ Head doorbell register to release Completion Queue entries to the controller and clear the associated interrupt.

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 all required Submission and Completion Queues prior to submitting commands to the controller. I/O Submission and Completion Queues are created using Admin commands defined in section 5.

7.2 Command Submission and Completion Mechanism (Informative)

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

7.2.1 Command Processing

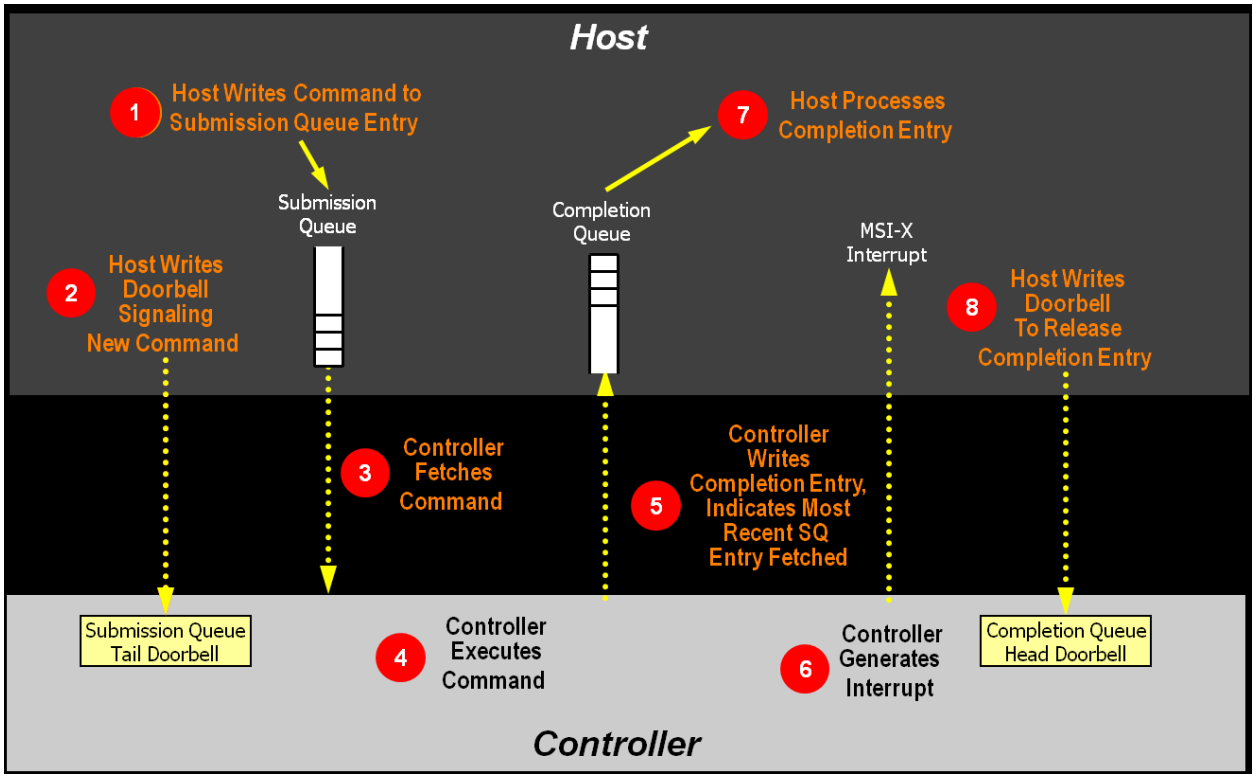
This section describes command submission and completion processing. Figure 251 shows the steps that are followed to submit and complete a command. The steps are:

1. The host places one or more commands for execution in the next free Submission Queue slot(s) in memory.
2. The host updates the Submission Queue Tail Doorbell register with the new value of the Submission Queue Tail entry pointer. This indicates to the controller that a new command(s) is submitted for processing.
3. The controller transfers the command(s) from in the Submission Queue slot(s) into the controller for future execution. Arbitration is the method used to determine the Submission Queue from which the controller starts processing the next candidate command(s), refer to section 4.11.
4. The controller then proceeds with execution of the next command(s). Commands may complete out of order (the order submitted or started execution).
5. After a command has completed execution, the controller places a completion queue entry in the next free slot in the associated Completion Queue. As part of the completion queue entry, the controller indicates the most recent Submission Queue entry that has been consumed by advancing the Submission Queue Head pointer in the completion entry. Each new completion

queue entry has a Phase Tag inverted from the previous entry to indicate to the host that this completion queue entry is a new entry.

6. The controller optionally generates an interrupt to the host to indicate that there is a new completion queue entry to consume and process. In the figure, this is shown as an MSI-X interrupt, however, it could also be a pin-based or MSI interrupt. Note that based on interrupt coalescing settings, an interrupt may or may not be generated for each new completion queue entry.
7. The host consumes and then processes the new completion queue entries in the Completion Queue. This includes taking any actions based on error conditions indicated. The host continues consuming and processing completion queue entries until it encounters a previously consumed entry with a Phase Tag inverted from the value of the current completion queue entries.
8. The host writes the Completion Queue Head Doorbell register to indicate that the completion queue entry has been consumed. The host may consume many entries before updating the associated Completion Queue Head Doorbell register.

Figure 251: Command Processing



7.2.2 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 (SQx) 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 SQx[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, CDW1.NSID, 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.
g. CDW10 – CDW15 are set to any command specific information.

2. Host software writes the corresponding Submission Queue doorbell register (SQxTDBL) to submit one or more commands for processing.

The write to the Submission Queue doorbell register triggers the controller to consume one or more new commands contained in the Submission Queue entry. The controller indicates the most recent SQ entry that has been consumed as part of reporting completions. Host software may use this information to determine when SQ slots may be re-used for new commands.

7.2.3 Processing Completed Commands

Host software processes the interrupt generated by the controller for command completion(s). If MSI-X or multiple message MSI is in use, then the interrupt vector implies the Completion Queue(s) with new command completions for the host to process. If pin-based interrupts or single message MSI interrupts are used, then host software interrogates the Completion Queue(s) to determine if new completion queue entries are present for the host to process.

Once the host software determines the Completion Queue (CQy) that generated the interrupt:

1. Host software reads a completion queue entry from the specified Completion Queue.
2. Host software processes the CQ entry to identify the Submission Queue entry that generated this completion. DW2.SQID indicates the Submission Queue ID and DW3.CID indicates the command that generated the completion.
3. DW3.SF indicates the status of the completion.
4. Host software indicates available Completion Queue slots by updating the corresponding Completion Queue Head doorbell register (CQyHDBL). By updating CQyHDBL, the associated interrupt is cleared.
5. If there were errors, noted in the DW3.SF field, host software performs error recovery actions (refer to section 10.1).

7.2.4 Command Related Resource Retirement

As part of reporting completions, the controller indicates the most recent Submission Queue entry that has been consumed. Submission Queue slots containing consumed Submission Queue entries are free and may be re-used by host software to submit new commands.

If a completion queue entry is posted for a command, then host software may re-use the associated PRP List(s) for that command and other resources (an exception is the PRP List for I/O Submission Queues and I/O Completion Queues).

7.2.5 Command Examples

7.2.5.1 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.

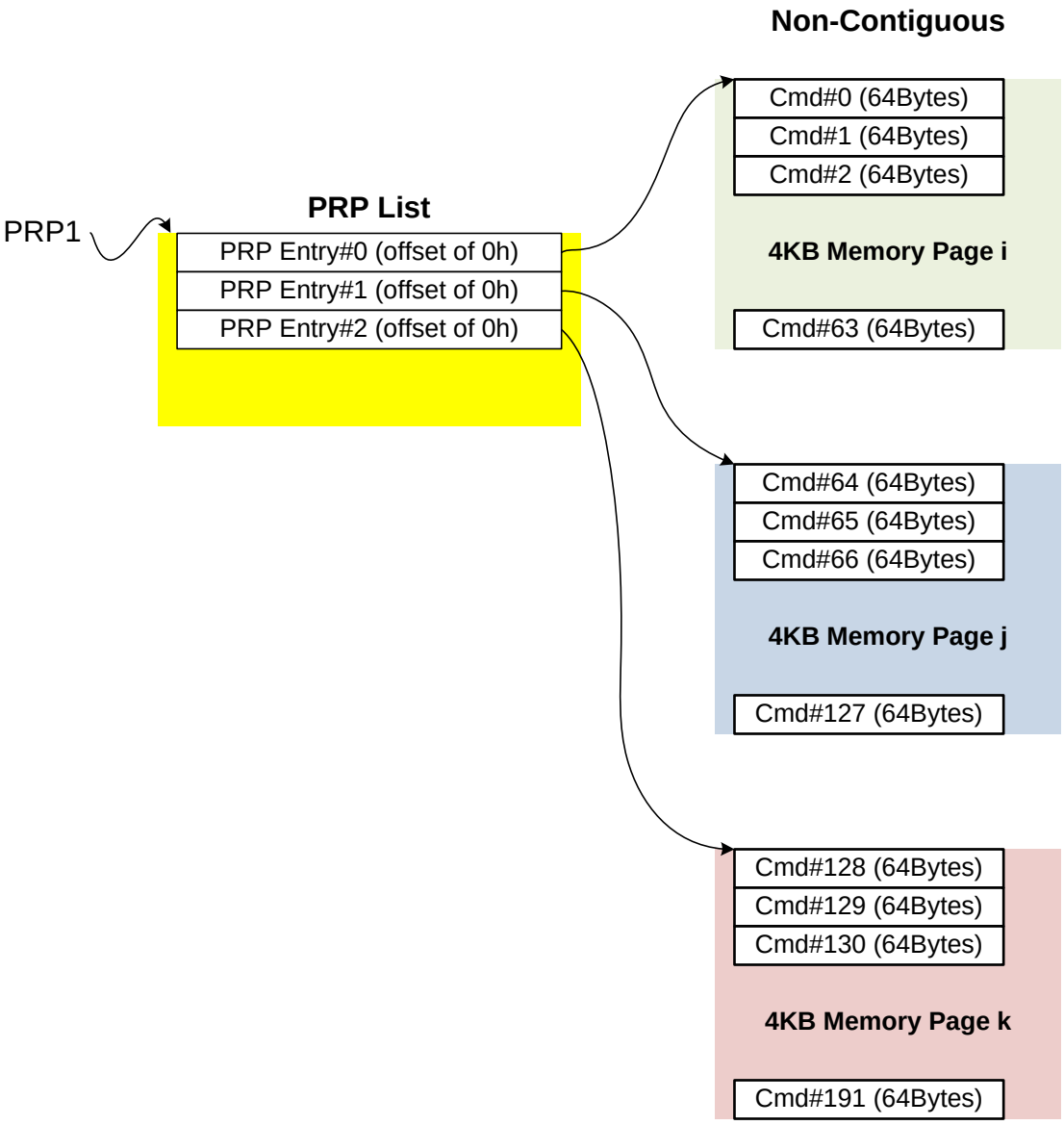
To create an I/O Submission Queue, host software builds a Create I/O Submission Queue command for the Admin Submission Queue. Host software builds the Create I/O Submission Queue command in the next free Admin Submission Queue command location. The attributes of the command are:

- CDW0.OPC is set to 01h.
- CDW0.FUSE is set to 00b indicating that this is not a fused operation.
- CDW0.CID is set to a free command identifier.
- CDW1.NSID is set 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 252 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, it is set to a value of 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.
- CDW11.PC is cleared to '0' indicating that the data buffer indicated by PRP1 is not physically contiguously.

After the command is built, host software submits the command for execution by writing the Admin Submission Queue doorbell (SQ0TDBL) to indicate to the controller that this command is available for processing.

Host software shall maintain the PRP List unmodified in host memory until the Submission Queue is deleted.

Figure 252: PRP List Describing I/O Submission Queue



7.2.5.2 Executing a Fused Operation

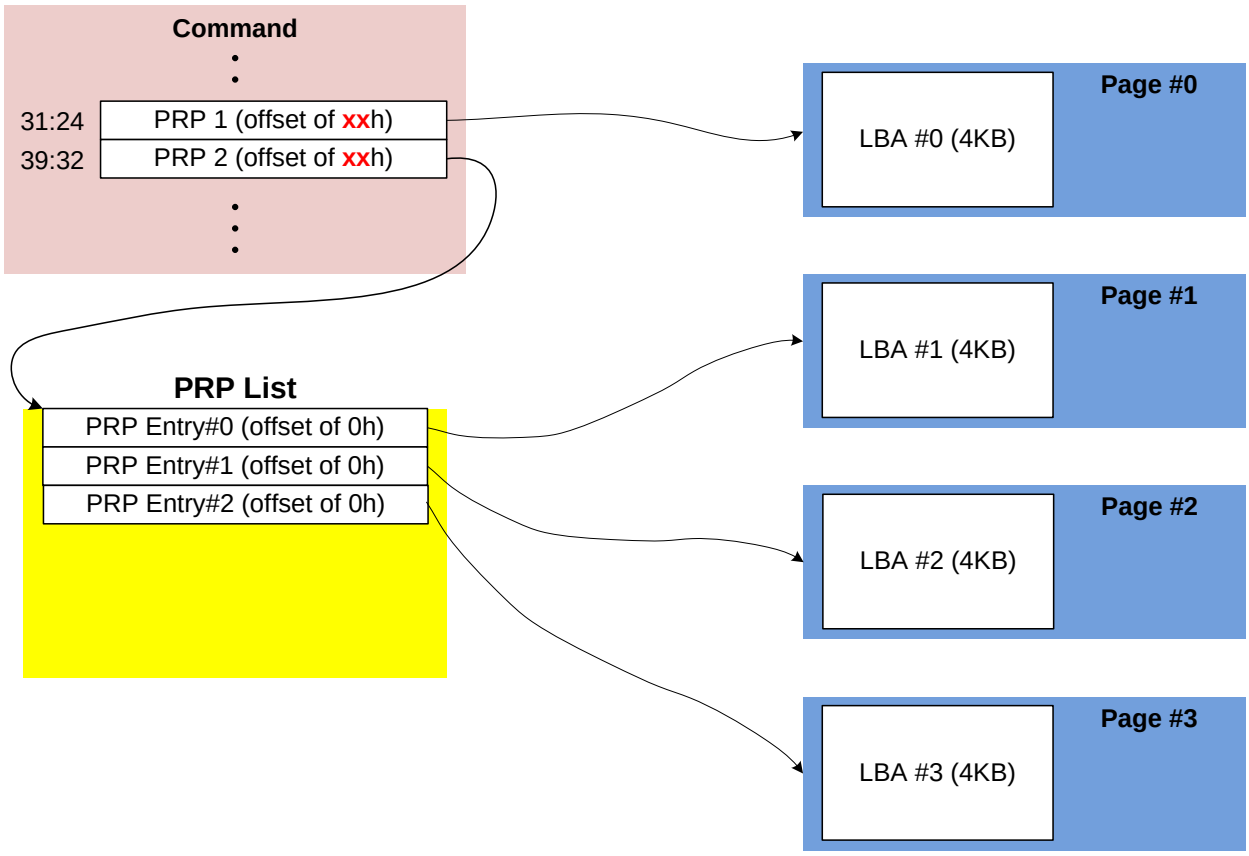
This example describes how host software creates and executes a fused command, specifically Compare and Write for a total of 16KB of data. In this case, there are two commands that are created. The first command is the Compare, referred to as CMD0. The second command is the Write, referred to as CMD1. In this case, end-to-end data protection is not enabled and the size of each logical block is 4KB.

To build commands for a fused operation, host software utilizes the next two available adjacent command locations in the appropriate I/O Submission Queue.

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.CDW1.NSID is set to 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 253 for a PRP List with three entries.
    - 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 DRAM 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 4KB each are to be compared against.
- CMD0.CDW14 is cleared to 0h since end-to-end protection is not enabled.
- CMD0.CDW15 is cleared to 0h since end-to-end protection is not enabled.

Figure 253: 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.CDW1.NSID is set to the appropriate namespace. This value shall be the same as CMD0.CDW1.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.
    - 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 DRAM 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 4KB 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.
- CMD1.CDW15 is cleared to 0h since end-to-end protection is not enabled.

After the commands are built, host software submits the commands for execution by writing the appropriate I/O Submission Queue doorbell (SQxTDBL) to indicate to the controller that these commands are submitted. Note that the doorbell write shall indicate both commands have been submitted at one time.

7.3 Resets

7.3.1 NVM Subsystem Reset

An NVM Subsystem Reset is initiated when:

- Power is applied to the NVM subsystem,
- A value of 4E564D65h (“NVMe”) is written to the NSSR.NSSRC field, 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 and a transition to the Detect LTSSM state by all PCI Express ports of the NVM subsystem.

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.

7.3.2 Controller Level Reset

There are five primary Controller Level Reset mechanisms:

- NVM Subsystem Reset
- Conventional Reset (PCI Express Hot, Warm, or Cold reset)
- PCI Express transaction layer Data Link Down status
- Function Level Reset (PCI reset)
- Controller Reset (CC.EN transitions from ‘1’ to ‘0’)

When any of the above resets occur, the following actions are performed:

- 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’.
- The Admin Queue registers (AQA, ASQ, or ACQ) are not reset as part of a controller reset. All other controller registers defined in section 3 and internal controller state are reset.

In all cases except a Controller Reset, the PCI register space is reset as defined by the PCI Express base specification. Refer to the PCI Express specification for further details.

To continue after a reset, the host shall:

- Update register 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.
- Proceed with normal I/O operations.

Note that all cases except a Controller Reset result in the controller immediately losing communication with the host. In these cases, the controller is unable to indicate any aborts or update any completion queue entries.

7.3.3 Queue Level

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 reset, 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 if desired.

The host should ensure that the appropriate I/O Submission Queue or I/O Completion Queue is idle before deleting it. 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). 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.

7.4 Queue Management

7.4.1 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) registers appropriately.
2. Submits a Set Features command with the Number of Queues attribute to request the desired 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.
3. Determines the maximum number of entries supported per queue (CAP.MQES) and whether the queues are required to be physically contiguous (CAP.CQR).
4. Creates the desired 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.
5. Creates the desired 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, the desired I/O Submission Queues and I/O Completion Queues have been setup and initialized and may be used to complete I/O commands.

7.4.2 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.

7.4.3 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 any commands desired to the associated I/O Submission Queue.

7.5 Interrupts

The interrupt architecture allows for efficient reporting of interrupts such that the host may service interrupts through the least amount of overhead.

The specification allows the controller to be configured to report interrupts in one of four modes. The four modes are: pin-based interrupt, single message MSI, multiple message MSI, and MSI-X. It is recommended that MSI-X be used whenever possible to enable higher performance, lower latency, and lower CPU utilization for processing interrupts.

Interrupt aggregation, also referred to as interrupt coalescing, mitigates host interrupt overhead by reducing the rate at which interrupt requests are generated by a controller. This reduced host overhead typically comes at the expense of increased latency. Rather than prescribe a specific interrupt aggregation algorithm, this specification defines the mechanisms a host may use to communicate desired interrupt aggregation parameters to a controller and leaves the specific interrupt aggregation algorithm used by a controller as vendor specific. Interrupts associated with the Admin Completion Queue should not be delayed.

The Aggregation Threshold field in the Interrupt Coalescing feature (refer to section 5.21.1.8) specifies the host desired minimum interrupt aggregation threshold on a per vector basis. This value defines the number of Completion Queue entries that when aggregated on a per interrupt vector basis reduces host interrupt processing overhead below a host determined threshold. This value is provided to the controller as a recommendation by the host and a controller is free to generate an interrupt before or after this aggregation threshold is achieved. The specific manner in which this value is used by the interrupt aggregation algorithm implemented by a controller is implementation specific.

The Aggregation Time field in the Interrupt Coalescing feature (refer to section 5.21.1.8) specifies the host desired maximum delay that a controller may apply to a Completion Queue entry before an interrupt is signaled to the host. This value is provided to the controller as a recommendation by the host and a controller is free to generate an interrupt before or after this aggregation time is achieved. A controller may apply this value on a per vector basis or across all vectors. The specific manner in which this value is used by the interrupt aggregation algorithm implemented by a controller is implementation specific.

Although support of the Get Features and Set Features commands associated with interrupt coalescing is required, the manner in which the Aggregation Threshold and Aggregation Time fields are used is implementation specific. For example, an implementation may ignore these fields and not implement interrupt coalescing.

7.5.1 Pin Based, Single MSI, and Multiple MSI Behavior

This is the mode of interrupt operation if any of the following conditions are met:

- Pin based interrupts are being used – MSI (MSICAP.MC.MSIE='0') and MSI-X are disabled
- Single MSI is being used – MSI is enabled (MSICAP.MC.MSIE='1'), MSICAP.MC.MME=0h, and MSI-X is disabled
- Multiple MSI is being used – Multiple-message MSI is enabled (MSICAP.MC.MSIE='1'), MSICAP.MC.MME is set to a value between 001b and 101b inclusive, and (MSICAP.MC.MME=1h) and MSI-X is disabled.

Within the controller there is an interrupt status register (IS) that is not visible to the host. In this mode, the IS register determines whether the PCI interrupt line shall be driven active or an MSI message shall be sent. Each bit in the IS register corresponds to an interrupt vector. The IS bit is set to '1' when the AND of the following conditions is true:

- There is one or more unacknowledged completion queue entries in a Completion Queue that utilizes this interrupt vector;
- The Completion Queue(s) with unacknowledged completion queue entries has interrupts enabled in the "Create I/O Completion Queue" command;
- The corresponding INTM bit exposed to the host is cleared to '0', indicating that the interrupt is not masked.

For single and multiple MSI, the INTM register masks interrupt delivery prior to MSI logic. As such, an interrupt on a vector masked by INTM does not cause the corresponding Pending bit to assert within the MSI Capability Structure.

If MSIs are not enabled, IS[0] being a one causes the PCI interrupt line to be active (electrical '0'). If MSIs are enabled, any change to the IS register that causes an unmasked status bit to transition from zero to one or clearing of a mask bit whose corresponding status bit is set shall cause an MSI to be sent. Therefore, while in wire mode, a single wire remains active, while in MSI mode, several messages may be sent, as each edge triggered event on a port shall cause a new message.

In order to clear an interrupt for a particular interrupt vector, host software shall acknowledge all completion queue entries for Completion Queues associated with the interrupt vector.

Status of IS Register	Pin-based Action	MSI Action
All bits '0' **Note:** May be caused by corresponding bit(s) in the INTM register being set to '1', masking the corresponding interrupt.	Wire inactive	No action
One or more bits set to '1' **Note:** May be caused by corresponding bit(s) in the INTM register being cleared to '0', unmasking the corresponding interrupt.	Wire active	New message sent
One or more bits set to '1', new bit gets set to '1'	Wire active	New message sent
One or more bits set to '1', some (but not all) bits in the IS register are cleared (i.e., host software acknowledges some of the associated completion queue entries)	Wire active	New message sent
One or more bits set to '1', all bits in the IS register are cleared (i.e., host software acknowledges all associated completion queue entries)	Wire inactive	No action

7.5.1.1 Host Software Interrupt Handling

It is recommended that host software utilize the Interrupt Mask Set and Interrupt Mask Clear (INTMS/INTMC) registers to efficiently handle interrupts when configured to use pin based or MSI messages. Specifically, within the interrupt service routine, host software should set the appropriate mask register bits to '1' to mask interrupts via the INTMS register. In the deferred procedure call, host software

should process all completion queue entries and acknowledge the completion queue entries have been processed by writing the associated CQyHDBL doorbell registers. When all completion queue entries have been processed, host software should unmask interrupts by clearing the appropriate mask register bits to ‘0 via the INTMC register.

It is recommended that the MSI interrupt vector associated with the CQ(s) being processed be masked during processing of completion queue entries within the CQ(s) to avoid spurious and/or lost interrupts. For single message or multiple message MSI, the INTMS and INTMC registers should be used to appropriately mask interrupts during completion queue entry processing.

7.5.1.1.1 Interrupt Example (Informative)

An example of the host software flow for processing interrupts is described in this section. This example assumes multiple message MSI is used and that interrupt vector 3 is associated with I/O Completion Queue 3.

1. The controller posts a completion queue entry to I/O Completion Queue 3. The controller sets IS[3] to ‘1’ in its internal IS register. The controller asserts an interrupt to the host.
2. The interrupt service routine (ISR) is triggered.
3. Host software scans all I/O Completion Queues associated with the asserted MSI vector to determine the location of new completion queue entries. In this case, a new completion queue entry has been posted to I/O Completion Queue 3.
4. Host software writes 08h to the INTMS register to mask interrupts for interrupt vector 3, which is associated with I/O Completion Queue 3.
5. The controller masks interrupt vector 3, based on the host write to the INTMS register.
6. Host software schedules a deferred procedure call (DPC) to process the completed command.
7. The deferred procedure call (DPC) is triggered.
8. Host software processes new completion queue entries for I/O Completion Queue 3, completing the associated commands to the OS. Host software updates CQyHDBL to acknowledge the processed completion queue entries and clear the interrupt associated with those completion queue entries. If all completion queue entries have been acknowledged by host software, the controller de-asserts interrupt vector 3.
9. Host software unmasks interrupt vector 3 by writing 08h to the INTMC register.

7.5.1.2 Differences Between Pin Based and MSI Interrupts

Single MSI is similar to the pin based interrupt behavior mode. The primary difference is the method of reporting the interrupt. Instead of communicating the interrupt through an INTx virtual wire, an MSI message is generated to the host. Unlike INTx virtual wire interrupts which are level sensitive, MSI interrupts are edge sensitive.

Pin based and single MSI only use one interrupt vector. Multiple MSI may use up to 32 interrupt vectors.

For multiple MSI, the controller advertises the number of MSI interrupt vectors it requests in the Multiple Message Capable (MMC) field in the Message Signaled Interrupt Message Control (MC) register. The MSICAP.MC.MMC field represents a power-of-2 wrapper on the number of requested vectors. For example, if three vectors are requested, then the MSICAP.MC.MMC field shall be ‘010’ (four vectors).

Multiple-message MSI allows completions to be aggregated on a per vector basis. If sufficient MSI vectors are allocated, each Completion Queue(s) may send its own interrupt message, as opposed to a single message for all completions.

7.5.2 MSI-X Based Behavior

This is the mode of interrupt operation if the MSI-X is being used – (multiple-message) MSI is disabled (MSICAP.MC.MSIE=’0’) and (MSICAP.MC.MME=0h) and MSI-X is enabled. This is the preferred interrupt behavior to use.

MSI-X, similar to multiple-message MSI, allows completions to be aggregated on a per vector basis. However, the maximum number of vectors is 2K. MSI-X also allows each interrupt to send a unique message data corresponding to the vector.

MSI-X allows completions to be aggregated on a per vector basis. Each Completion Queue(s) may send its own interrupt message, as opposed to a single message for all completions.

When generating an MSI-X message, the following checks occur before generating the message:

- The function mask bit in the MSI-X Message Control register is not set to '1'
- The corresponding vector mask in the MSI-X table structure is not set to '1'

If either of the masks are set, the corresponding pending bit in the MSI-X PBA structure is set to '1' to indicate that an interrupt is pending for that vector. The MSI for that vector is later generated when both the mask bits are reset to '0'.

It is recommended that the interrupt vector associated with the CQ(s) being processed be masked during processing of completion queue entries within the CQ(s) to avoid spurious and/or lost interrupts. The interrupt mask table defined as part of MSI-X should be used to mask interrupts.

7.6 Controller Initialization and Shutdown Processing

This section describes the recommended procedure for initializing the controller and for shutdown processing prior to a power-off condition.

7.6.1 Initialization

The host should perform the following actions in sequence to initialize the controller to begin executing commands:

1. Set the PCI and PCI Express registers described in section 2 appropriately based on the system configuration. This includes configuration of power management features. A single interrupt (e.g. pin-based, single-MSI, or single MSI-X) should be used until the number of I/O Queues is determined.
2. The host waits for the controller to indicate that any previous reset is complete by waiting for CSTS.RDY to become '0.'
3. The Admin Queue should be configured. The Admin Queue is configured by setting the Admin Queue Attributes (AQA), Admin Submission Queue Base Address (ASQ), and Admin Completion Queue Base Address (ACQ) to appropriate values.
4. The controller settings should be configured. Specifically:
    a. The arbitration mechanism should be selected in CC.AMS.
    b. The memory page size should be initialized in CC.MPS.
    c. The I/O Command Set that is to be used should be selected in CC.CSS.
5. The controller should be enabled by setting CC.EN to '1'.
6. The host should wait for the controller to indicate it is ready to process commands. The controller is ready to process commands when CSTS.RDY is set to '1'.
7. The host should determine the configuration of the controller by issuing the Identify command, specifying the Controller data structure. The host should then determine the configuration of each namespace by issuing the Identify command for each namespace, specifying the Namespace data structure.
8. 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 MSI and/or MSI-X registers should be configured.
9. 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.

10. 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.
11. If the host desires asynchronous notification of optional events, the host should issue a Set Features command specifying the events to enable. If the host desires 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 it is ready (i.e., CSTS.RDY is set to '1').

After performing these steps, the controller may be used for I/O commands.

For exit of the D3 power state, the initialization steps outlined should be followed. In this case, the number of I/O Submission Queues and I/O Completion Queues shall not change, thus step 7 of the initialization sequence is optional.

7.6.1.1 Software Progress Marker

The Software Progress Marker feature, defined in section 5.21.1.18, indicates the number of times pre-boot software has loaded prior to the OS successfully loading. If the pre-boot software load count becomes large, it may indicate there are issues with cached data within the NVM since the OS driver software has not set this field to 0h recently. In this case, the OS driver software may choose to use the NVM more conservatively (e.g., not utilize cached data).

The Software Progress Marker should be updated by both Pre-boot and OS driver software as part of completing initialization.

7.6.2 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 shutdown:

1. Stop submitting any new I/O commands to the controller and allow any outstanding commands to complete.
2. 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.
3. The host should delete all I/O Completion Queues, using the Delete I/O Completion Queue command.
4. The host should set the Shutdown Notification (CC.SHN) field to 01b to indicate a normal shutdown operation. The controller indicates when shutdown processing is completed by updating the Shutdown Status (CSTS.SHST) field to 10b.

For entry to the D3 power state, the shutdown steps outlined for a normal shutdown should be followed.

The host should perform the following actions in sequence for an abrupt shutdown:

1. Stop submitting any new I/O commands to the controller.
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.

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.

It is safe to power off the controller when CSTS.SHST indicates 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 a shutdown operation, a Controller Reset (CC.EN cleared from '1' to '0') is required. The initialization sequence should then be executed.

It is an implementation choice whether the host aborts all outstanding commands to the Admin Queue prior to the shutdown. The only commands that should be outstanding to the Admin Queue at shutdown are Asynchronous Event Request commands.

7.7 Asynchronous Event Request Host Software Recommendations (Informative)

This section describes the recommended host software procedure for Asynchronous Event Requests.

The host sends *n* Asynchronous Event Request commands (refer to section 7.6.1, step 11). When an Asynchronous Event Request completes (providing Event Type, Event Information, and Log Page details):

1. If the event(s) in the reported Log Page may be disabled with the Asynchronous Event Configuration feature (refer to section 5.21.1.11), 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.
2. 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.
3. 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.
4. Host software should issue an Asynchronous Event Request command to the controller (restoring to *n* the number of these commands outstanding).
5. 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.

7.8 Feature Values

The Get Features command, defined in section 5.13, and Set Features command, defined in section 5.21, 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 field of the Identify Controller data structure in Figure 109 then for each Feature, there are three settings: default, saveable, and current. If bit 4 is cleared to '0' in the Optional NVM Command Support field of the Identify Controller data structure in Figure 109 then the controller only supports a current and default value for each Feature. In this case, the current value may be persistent across power states based on the information specified in Figure 134 and Figure 135.

The default value for each Feature is vendor specific and set by the manufacturer unless otherwise specified; it is not changeable. The saveable value is the value that the Feature has after a power on or reset event. The controller may not support a saveable value for a Feature; this is discovered by using the 'supported capabilities' value in the Select field in Get Features. If the controller does not support a saveable value for a Feature, then the default value is used after a power on or reset event. The current value is the value actively in use by the controller for a Feature after a Set Features command completes.

Set Features may be used to modify the saveable and current value for a Feature. Get Features may be used to read the default, saveable, and current value for a Feature. If the controller does not support a saveable value for a Feature, then the default value is returned for the saveable value in Get Features.

Feature settings may apply to the entire controller (and all associated namespaces) or may apply to each namespace individually. To change or retrieve a value that applies to the controller and all associated namespaces, host software sets CDW1.NSID to 0h or FFFFFFFFh in the Set Features or Get Features command. Features that are not namespace specific shall have the CDW1.NSID field set to 0h.

To change or retrieve a value that applies to a specific namespace, host software sets CDW1.NSID to the identifier of that namespace in the Set Features or Get Features command. If host software specifies a valid CDW1.NSID value that is not 0h or FFFFFFFFh and the Feature is not namespace specific, then a Set Features command returns the Feature Not Namespace Specific status code, whereas a Get Features command returns the Feature value that applies to the entire controller.

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 134 and Figure 135. 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 of Invalid Field in Command.

7.9 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)
  - colon (':'=U+003a)
- The maximum name is 223 bytes in length.
- 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 readable string to describe the host or NVM subsystem. This format consists of:

- The string "nqn."
- A date code, in "yyyy-mm." format. This date shall be during a time 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 reverse domain name of the naming authority that is creating the NQN.
- 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 following are examples of NVMe Qualified Names that may be generated by "Example NVMe, Inc."

- nqn.2014-08.com.example:nvme:nvm-subsystem-sn-d78432
- 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 desire for a human readable string. This format consists of:

- The string “nqn.”
- The string “2014-08.org.nvmexpress:uuid:”.
- 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:

- nqn.2014-08.org.nvmexpress:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6

7.10 Identifier Format and Layout (Informative)

This section provides guidance for proper implementation of various identifiers defined in the Identify Controller and Identify Namespace data structures.

7.10.1 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
- SSVID = 1234h

Byte	00	01	02	03
Value	CDh	ABh	34h	12h

7.10.2 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”
- MN = “M2”

Byte	04	05	06	23 - 07	24	25	63 - 26
Value	53h (‘S’)	4Eh (‘N’)	31h (‘1’)	20h (‘ ‘)	4Dh (‘M’)	32h (‘2’)	20h (‘ ‘)

7.10.3 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

Byte	73	74	75
Value	EFh	CDh	ABh

7.10.4 IEEE Extended Unique Identifier (EUI64)

The IEEE Extended Unique Identifier (EUI64, bytes 127:120) is defined in the Identify Namespace data structure. A tutorial is available at https://standards.ieee.org/develop/regauth/tut/eui64.pdf. 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:

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 7.10.3.

Example:

- OUI Identifier = ABCDEFh
- Extension Identifier = 0123456789h

Byte	120	121	122	123	124	125
Value	ABh	CDh	EFh	01h	23h	45h
Field	OUI			Extension Identifier		

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.

Byte	0	1	2	3	4	5	6	7
EUI64	OUI			Extension Identifier				
WWN (NAA = 5)	5h	OUI		Vendor Specific Identifier				

7.10.5 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 7.10.3.

Example:

- OUI Identifier = ABCDEFh
- Extension Identifier = 0123456789h
- Vendor Specific Extension Identifier = FEDCBA9876543210h

Byte	104	105	106	107	108	109
Value	FEh	DCh	BAh	98h	76h	54h
Field	Vendor Specific Extension Identifier					

Byte	110	111	112	113	114	115
Value	32h	10h	ABh	CDh	EFh	01h
Field	VSP Ex ID (cont)		OUI			Ex ID

Byte	116	117	118	119
Value	23h	45h	67h	89h
Field	Extension Identifier (cont)			

The NGUID format is similar to the World Wide Name (WWN) format as IEEE Registered Extended designator (NAA = 6) as shown below.

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							

7.11 Unique Identifier

The NVM Subsystem NVMe Qualified Name specified in the Identify Controller data structure 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 this specification that does not include the NVM Subsystem NQN, then the PCI Vendor ID, Serial Number, and Model Number fields in the Identify Controller and the NQN Starting String “nqn.2014.08.org.nvmexpress:” may be combined to form a globally unique value that identifies the NVM subsystem (e.g., for host software that uses NQNs). The method shown in Figure 254 should be used 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 254: 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.
255:95	**Padding (PAD):** Contains spaces (ASCII character 20h).

An NVM subsystem may contain multiple controllers. All of the controllers that make up an 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 contains the IEEE Extended Unique Identifier (EUI64) and the Namespace Globally Unique Identifier (NGUID) fields. EUI64 is an 8-byte EUI-64 identifier and NGUID is a 16-byte identifier based on EUI-64. When creating a namespace, the controller specifies a globally unique value in the EUI64 or NGUID field (the controller may optionally specify a globally unique value in both fields). In cases where the 64-bit EUI64 field is unable to ensure a globally unique namespace identifier, the EUI64 field shall be cleared to 0h. When not implemented, these fields contain a value of 0h. A controller may reuse a non-zero NGUID or EUI64 value for a new namespace after the original namespace using the value has been deleted. If bit 3 in NSFEAT 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.

7.12 Keep Alive

The Keep Alive feature (refer to section 5.21.1.15) is used by the host to determine that the controller is operational and 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 execute commands. The controller indicates the granularity of the Keep Alive Timer in the Identify Controller data structure.

The Keep Alive is a watchdog timer intended to detect a malfunctioning connection, controller, or host. 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 only for an enabled controller, i.e., the Keep Alive timer is active if:

- CC.EN is set to '1' and CSTS.RDY is set to '1'; and
- CC.SHN is cleared to '0' and CSTS.SHST is cleared to '0'.

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) shall initialize the Keep Alive timer to the Keep Alive Timeout value.

The host may consider a Keep Alive Timeout to have occurred when it does not receive the completion of the Keep Alive command 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.

When a Keep Alive Timer for the Admin Queue expires:

- the controller records an Error Information Log Entry with the status code Keep Alive Timeout Expired and sets the Controller Fatal Status (CSTS.CFS) bit to '1'; and
- the host assumes all outstanding commands are not completed and need to be re-issued.

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 it is received.

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 Keep Alive is required.

NVMe Transports that do not detect a connection loss in a timely manner shall require that the Keep Alive be enabled. If a command attempts to disable Keep Alive by setting the timeout value to 0h or to a value that exceeds the maximum allowed by the associated NVMe Transport binding specification, a status value of Keep Alive Invalid shall be returned. If a command sets the timeout value to a value that is smaller than the minimum supported by the NVMe Transport or specific implementation, then the controller rounds up the timeout to the minimum.

7.12.1 NVMe over PCIe

Keep Alive is not required for NVMe over PCIe. The PCIe Transport does not impose any limitations on the minimum and maximum Keep Alive Timeout value.

7.13 Updating Controller Doorbell Registers using a Shadow Doorbell Buffer

7.13.1 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 5.7).

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 register. 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 register 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 register).

7.13.2 Example Algorithm for Controller Doorbell Register Updates (Informative)

Host software may use modular arithmetic where the modulus is the queue depth to decide if the controller doorbell register should be updated, specifically:

- Compute *X* as the new doorbell value minus the corresponding EventIdx value, modulo queue depth.
- 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 register should be updated with the new doorbell value.

8 Features

8.1 Firmware Update Process

The process for a firmware update to be activated by a reset is:

1. The host issues a Firmware Image Download command to download the firmware image to the controller. There may be multiple portions of the firmware image to download, thus the offset for each portion of the firmware image being downloaded 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 the controller, the next step is for the host to submit a Firmware Commit command. The Firmware Commit command verifies that the last firmware image downloaded is valid and commits that 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 5.11.1).
4. After the reset has completed, host software re-initializes the controller. This includes re-allocating I/O Submission and Completion Queues. Refer to section 7.6.1.

The process for a firmware update to be activated without a reset is:

1. The host issues a Firmware Image Download command to download the firmware image to the controller. There may be multiple portions of the firmware image to download, thus the offset for each portion of the firmware image being downloaded 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 with a Commit Action of 011b which specifies that the image should be activated immediately without reset. The downloaded image should replace the image in the firmware slot. If no image was downloaded since the last reset or Firmware Commit command, (i.e., the first step was skipped), then the controller shall verify and activate the image in the specified slot. If the controller starts to activate the firmware, any controllers affected by the new firmware send a Firmware Activation Starting asynchronous event to the host if Firmware Activation Notices are enabled (refer to Figure 148).
    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 reset is required to activate this firmware, then the controller reports an error of Firmware Activation Requires Reset and the image is applied at the next reset.

c. 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 image activated using a reset.

If a D3 cold condition occurs during the firmware activation process, the controller may resume operation with either the old or new firmware.

If the firmware is not able to be successfully loaded, then the controller shall revert to the previously active firmware image or the baseline read-only firmware image, if available, and indicate the failure as an asynchronous event with a Firmware Image Load Error.

Host software shall not update multiple firmware images simultaneously. A firmware image shall be committed to the firmware slot using Firmware Commit before downloading additional firmware images. If the controller does not receive a Firmware Commit command, then it shall delete the portion(s) of the new image in the case of a reset.

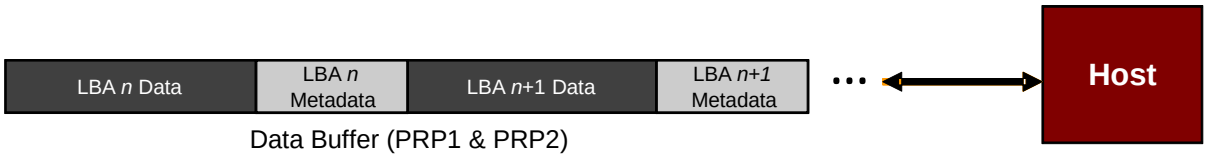
8.2 Metadata Handling

The controller may support metadata per logical block. Metadata is additional data allocated on a per logical block basis. There is no requirement for how the host makes use of the metadata area. One of the most common usages for metadata is to convey end-to-end protection information.

The metadata may be transferred by the controller to or from the host in one of two ways. The mechanism used is selected when the namespace is formatted.

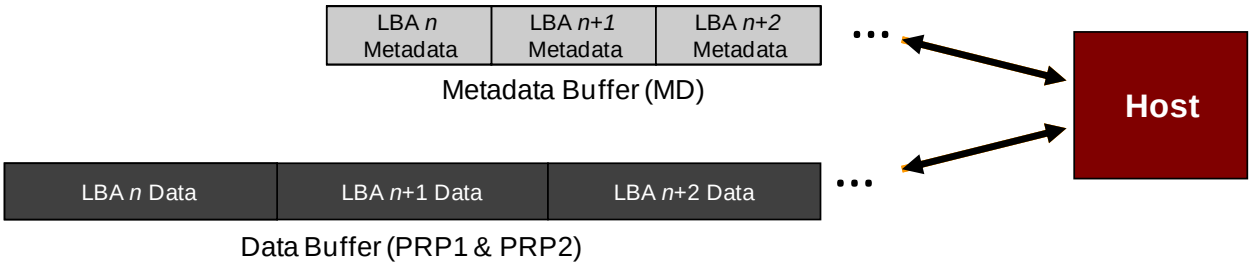
The first mechanism for transferring the metadata is as a contiguous part of the logical block that it is associated with. The metadata is transferred at the end of the associated logical block, forming an extended logical block. This mechanism is illustrated in Figure 255. In this case, both the logical block data and logical block metadata are pointed to by the PRP1 and PRP2 pointers (or SGL Entry 1 if SGLs are used).

Figure 255: Metadata – Contiguous with LBA Data, Forming Extended LBA



The second mechanism for transferring the metadata is as a separate buffer of data. This mechanism is illustrated in Figure 256. In this case, the metadata is pointed to with the Metadata Pointer, while the logical block data is pointed to by the Data Pointer. When a command uses PRPs for the metadata in the command, the metadata is required to be physically contiguous. When a command uses SGLs for the metadata in the command, the metadata is not required to be physically contiguous.

Figure 256: Metadata – Transferred as Separate Buffer



One of the transfer mechanisms shall be selected for each namespace when it is formatted; transferring a portion of metadata with one mechanism and a portion with the other mechanism is not supported.

If end-to-end data protection is used, then the Protection Information field for each logical block is contained in the metadata.

8.3 End-to-end Data Protection (Optional)

To provide robust data protection from the application to the NVM media and back to the application itself, end-to-end data protection may be used. If this optional mechanism is enabled, then additional protection information (e.g. CRC) is added to the logical block that may be evaluated by the controller and/or host software to determine the integrity of the logical block. This additional protection information, if present, is either the first eight bytes of metadata or the last eight bytes of metadata, based on the format of the namespace. For metadata formats with more than eight bytes, if the protection information is contained within the first eight bytes of metadata, then the CRC does not cover any metadata bytes. For metadata formats with more than eight bytes, if the protection information is contained within the last eight bytes of metadata, then the CRC covers all metadata bytes up to but excluding these last eight bytes. As described in section 8.2, metadata and hence this protection information may be configured to be contiguous with the logical block data or stored in a separate buffer.

The most commonly used data protection mechanisms in Enterprise implementations are SCSI Protection Information, commonly known as Data Integrity Field (DIF), and the Data Integrity Extension (DIX). The primary difference between these two mechanisms is the location of the protection information. In DIF, the protection information is contiguous with the logical block data and creates an extended logical block, while in DIX, the protection information is stored in a separate buffer. The end-to-end data protection mechanism defined by this specification is functionally compatible with both DIF and DIX. DIF functionality is achieved by configuring the metadata to be contiguous with logical block data (as shown in Figure 255), while DIX functionality is achieved by configuring the metadata and data to be in separate buffers (as shown in Figure 256).

NVM Express supports the same end-to-end protection types as DIF. The type of end-to-end data protection (Type 1, Type 2, or Type 3) is selected when a namespace is formatted and is reported in the Identify Namespace data structure.

The Protection Information format is shown in Figure 257 and is contained in the metadata associated with each logical block. The Guard field contains a CRC-16 computed over the logical block data. In addition to a CRC-16, DIX also specifies an optional IP checksum that is not supported by NVM Express. The Application Tag is an opaque data field not interpreted by the controller and that may be used to disable checking of protection information. The Reference Tag associates logical block data with an address and protects against misdirected or out-of-order logical block transfer. Like the Application Tag, the Reference Tag may also be used to disable checking of protection information.

Figure 257: Protection Information Format

8.3.1 The PRACT Bit

The protection information processing performed as a side effect of Read and Write commands is controlled by the Protection Information Action (PRACT) bit in the command.

8.3.1.1 Protection Information and Write Commands

Figure 258 provides some examples of the protection information processing that may occur as a side effect of a Write command.

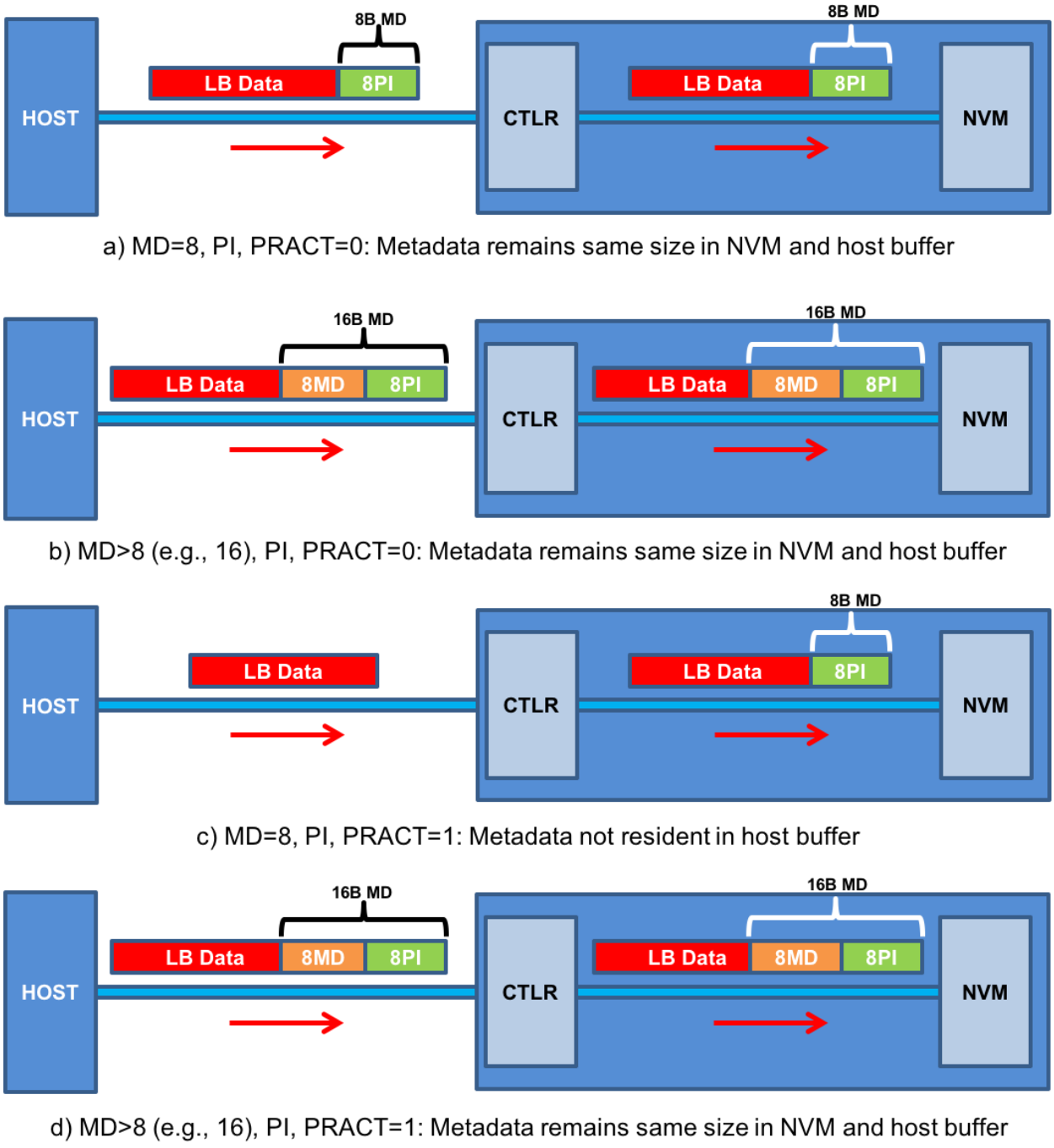
If the namespace is not formatted with end-to-end data protection, then logical block data and metadata is transferred from the host to the NVM with no protection information related processing by the controller.

If the namespace is formatted with protection information and the PRACT bit is cleared to ‘0’, then logical block data and metadata, which contains the protection information and may contain additional metadata, are transferred from the host buffer to NVM (i.e., the metadata field remains the same size in the NVM and the host buffer). As the logical block data and metadata passes through the controller, the protection information is checked. If a protection information check error is detected, the command completes with the status code of the error detected (i.e., End-to-end Guard Check, End-to-end Application Tag Check or End-to-end Reference Tag Check).

If the namespace is formatted with protection information and the PRACT bit is set to ‘1’, then:

1. If the namespace is formatted with Metadata Size equal to 8 (refer to Figure 115), then the logical block data is transferred from the host buffer to the controller. As the logical block data passes through the controller, the controller generates and appends protection information to the end of the logical block data, and the logical block data and protection information are written to NVM (i.e., the metadata is not resident within the host buffer).
2. If the namespace is formatted with Metadata Size greater than 8, then the logical block data and the metadata are transferred from the host buffer to the controller. As the metadata passes through the controller, the controller overwrites the protection information portion of the metadata. The logical block data and metadata are written to the NVM (i.e., the metadata field remains the same size in the NVM and the host buffer). The location of the protection information within the metadata is configured when the namespace is formatted (refer to the DPS field in Figure 114).

Figure 258: Write Command Protection Information Processing



NOTE: In cases (b) and (d) the Protection Information could be before or after the 8 bytes of metadata.

8.3.1.2 The PRACT Bit and Read Commands

Figure 259 provides some examples of the protection information processing that may occur as a side effect of Read command processing.

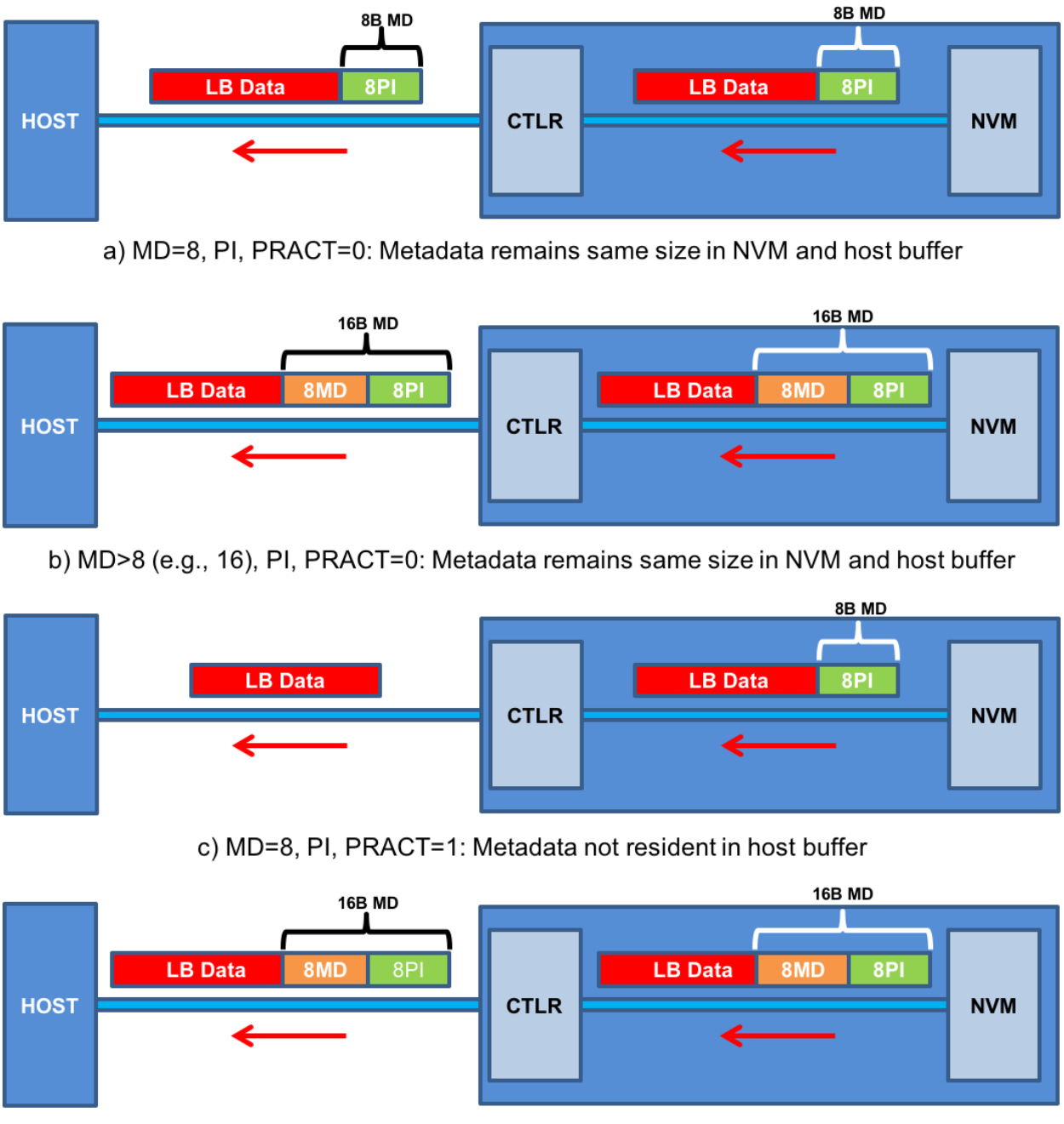
If the namespace is formatted with protection information and the PRACT bit is cleared to '0', then the logical block data and metadata, which in this case contains the protection information and possibly additional host metadata, is transferred by the controller from the NVM to the host buffer (i.e., the metadata field remains the same size in the NVM and the host buffer). As the logical block data and metadata pass through the controller, the protection information within the metadata is checked. If a protection information check error is detected, the command completes with the status code of the error detected (i.e., End-to-end Guard Check, End-to-end Application Tag Check or End-to-end Reference Tag Check).

If the namespace is formatted with protection information and the PRACT bit is set to '1', then:

a) if the namespace is formatted with Metadata Size equal to 8 (refer to Figure 115), the logical block data and metadata (which in this case is, by definition, the protection information)~~,~~ is read from the NVM by the controller. As the logical block and metadata pass through the controller, the protection information is checked. If a protection information check error is detected, the command completes with the status code of the error detected (i.e., End-to-end Guard Check, End-to-end Application Tag Check or End-to-end Reference Tag Check). After processing the protection information, the controller strips it and returns the logical block data to the host (i.e., the metadata is not resident within the host buffer);

b) if the namespace is formatted with Metadata Size greater than 8, the logical block data and the metadata, which in this case contains the protection information and additional host formatted metadata, is read from the NVM by the controller. As the logical block and metadata pass through the controller, the protection information embedded within the metadata is checked. If a protection information check error is detected, the command completes with the status code of the error detected (i.e., End-to-end Guard Check, End-to-end Application Tag Check or End-to-end Reference Tag Check). After processing the protection information, the controller passes the logical block data and metadata, with the embedded protection information unchanged, to the host (i.e., the metadata field remains the same size in the NVM as within the host buffer).

Figure 259: Read Command Protection Information Processing



a) MD=8, PI, PRACT=0: Metadata remains same size in NVM and host buffer

b) MD>8 (e.g., 16), PI, PRACT=0: Metadata remains same size in NVM and host buffer

c) MD=8, PI, PRACT=1: Metadata not resident in host buffer

d) MD>8 (e.g., 16), PI, PRACT=1: Metadata remains same size in NVM and host buffer

NOTE: In cases (b) and (d) the PI could be before or after the 8 bytes of metadata.

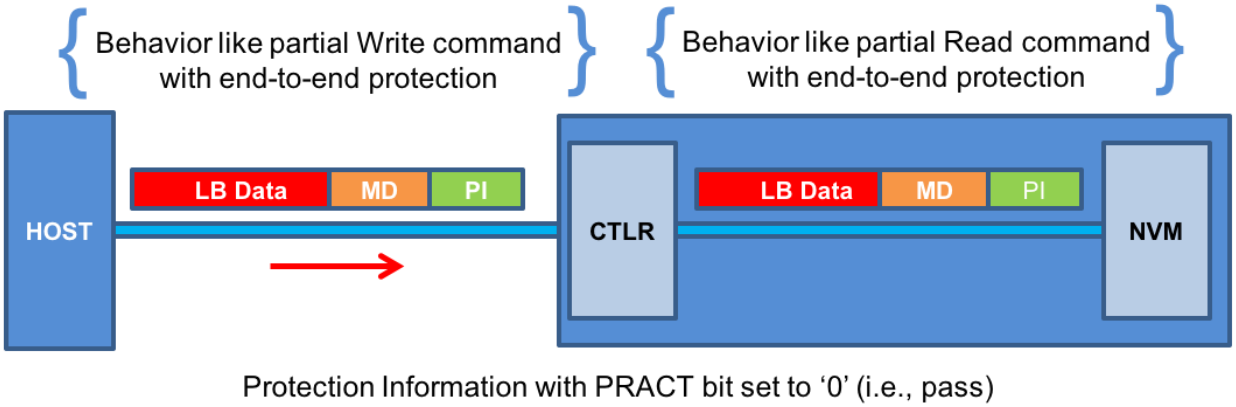
8.3.1.3 Protection Information for Fused Operations

Protection processing for fused operations is the same as those for the individual commands that make up the fused operation.

8.3.1.4 Protection Checking with the Compare command

Figure 260 illustrates the protection information processing that may occur as a side effect of Compare command processing. Compare command processing parallels both Write and Read commands. The controller checks the protection information contained in the command and the protection information read from the NVM.

Figure 260: Protection Information Processing for Compare



8.3.1.5 Control of Protection Information Checking - PRCHK

Checking of protection information consists of the following operations performed by the controller. If bit 2 of the Protection Information Check (PRCHK) field of the command is set to '1', then the controller compares the protection information Guard field to the CRC-16 computed over the logical block data. If bit 1 of the PRCHK field is set to '1', then the controller compares unmasked bits in the protection information Application Tag field to the Logical Block Application Tag (LBAT) field in the command. A bit in the protection information Application Tag field is masked if the corresponding bit is cleared to '0' in the Logical Block Application Tag Mask (LBATM) field of the command.

For Type 1 protection, if bit 0 of the PRCHK field is set to '1', then the controller compares the protection information Reference Tag field to the computed reference tag. The value of the computed reference tag for the first LBA of the command is the value contained in the Initial Logical Block Reference Tag (ILBRT) or Expected Initial Logical Block Reference Tag (EILBRT) field in the command. If the namespace is formatted for Type 1 or Type 2 protection, the computed reference tag is incremented for each subsequent logical block. If the namespace is formatted for Type 3 protection, the reference tag for each subsequent logic block remains the same as the initial reference tag. Unlike SCSI Protection Information Type 1 protection which implicitly uses the least significant four bytes of the LBA, the controller always uses the ILBRT or EILBRT field and requires host software to initialize the ILBRT or EILBRT field to the least significant four bytes of the LBA when Type 1 protection is used. In Type 1 protection, the controller should check the ILBRT or EILBRT field; if there is any miscompare, the command completes with an error of Invalid Protection Information. If the ILBRT or EILBRT field does not match the least significant four bytes of the LBA, then the controller completes the command with an Invalid Protection Information status code.

For Type 2 protection, if bit 0 of the PRCHK field is set to '1', then the controller compares the protection information Reference Tag field from each logical block to the computed reference tag. The computed reference tag is incremented for each subsequent logical block. The value of the computed reference tag for the first LBA of the command is the value contained in the ILBRT or EILBRT field in the command. Host software may set the ILBRT and EILBRT fields to any value.

For Type 3 protection, if bit 0 of the PRCHK field is set to '1', then the command may be aborted with status Invalid Field in Command. The controller may ignore the ILBRT and EILBRT fields when Type 3 protection is used because the computed reference tag remains unchanged.

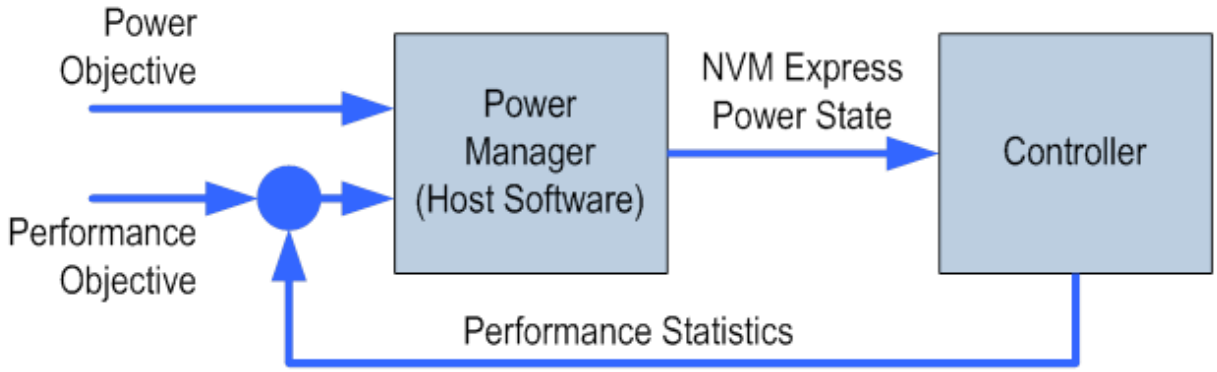
Protection checking may be disabled as a side effect of the value of the protection information Application Tag and Reference Tag fields regardless of the state of the PRCHK field in the command. If the namespace is formatted for Type 1 or Type 2 protection, then all protection information checks are disabled regardless of the state of the PRCHK field when the protection information Application Tag has a value of FFFFh. If the namespace is formatted for Type 3 protection, then all protection information checks are disabled regardless of the state of the PRCHK field when the protection information Application Tag has a value of FFFFh and the protection information Reference Tag has a value of FFFFFFFFh.

Inserted protection information consists of the computed CRC-16 in the Guard field, the LBAT field value in the Application Tag, and the computed reference tag in the Reference Tag field.

8.4 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 261 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 261: 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. A controller shall support at least one power state and may optionally support up to a total of 32 power states. Power states are 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 113). The descriptors for all implemented power states may be viewed as forming a table as shown in Figure 262 for a controller with seven implemented power states. Note that Figure 262 is illustrative and does not include all fields in the power state descriptor. The Maximum Power (MP) field indicates the maximum power that may be consumed in that state. Refer to the appropriate form factor specification for power measurement methodologies for that form factor. 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.

Figure 262: Example Power State Descriptor Table

Power State	Maximum Power (MP)	Entry Latency (ENTLAT)	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	5000 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 (i.e., there are no pending commands register 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 8.4.3). 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 (ENTLAT) field in the power management descriptor indicates the maximum amount of time in microseconds that it takes to enter that power state and the Exit Latency (EXLAT) field indicates the maximum amount of time in microseconds that it takes 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 need not have increasing relative performance values).

A lower relative performance value indicates better performance (e.g., higher throughput or lower latency). For example, in Figure 262 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 form factor specification used by the PCI Express SSD. The host shall never select a power state that consumes more power than the PCI Express slot power limit control value expressed by the Captured Slot Power Limit Value (CSPLV) and Captured Slot Power Limit Scale (CSPLS) fields of the PCI Express Device Capabilities (PXDCAP) register. Hosts that do not dynamically manage power should set the power state to the lowest numbered state that satisfies the PCI Express slot power limit control value.

If a controller implements the PCI Express Dynamic Power Allocation (DPA) capability and it is enabled (i.e., the Substate Control Enable bit is set), then the maximum power that may be consumed by the NVM subsystem is equal to the minimum value specified by the DPA substate or the NVM Express power state, whichever is lower.

8.4.1 Non-Operational Power States

A power state may be a non-operational power state, as indicated by Non-Operational State (NOPS) field in Figure 113. In a non-operational power state, memory-mapped I/O (MMIO) accesses, configuration register accesses and Admin Queue commands are serviced. 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 when an I/O Submission Queue Tail Doorbell is written.

While in a non-operational state, a controller may exceed the power advertised by the state for the following purposes:

- servicing a memory-mapped I/O (MMIO) or configuration register access;
- processing a command submitted to the Admin Submission Queue (such as Device Self-test command); or
- if Non-Operational Power State Permissive Mode is supported and enabled, executing controller initiated background operations (refer to section 5.21.1.17).

For all of the cases in the preceding paragraph, the controller shall:

- logically remain in the current non-operational power state unless an IO command is received or if an explicit transition is requested by a Set Features command with the Power Management identifier; and
- not exceed the maximum power advertised for the most recent operational power state.

8.4.2 Autonomous Power State Transitions

The controller may support autonomous power state transitions, as indicated in the Identify Controller data structure in Figure 109. 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 of Invalid Field in Command. Refer to section 8.4.1 for more details.

8.4.3 NVM Subsystem Workloads

The workload values described in this section may specify a workload hint in the Power Management Feature (refer to section 5.21.1.2) 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 263.

Figure 263: 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 1MB 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 128KB 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 – 111b	Reserved

8.4.4 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 109. 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 measured from the time power is applied until the controller is able to complete an I/O command. 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 then sends a 4KB read operation.

The RTD3 Entry Latency is measured from the time CC.SHN is set to 01b by host software until CC.SHST is set to 10b by the controller. When CC.SHST is set to 10b, it is safe for host software to remove power from the controller.

8.4.5 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 109. 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 5.21.1.16) 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 109.

The Thermal Management Temperature 1 specifies that if the Composite Temperature (refer to Figure 94) is:

a) at or above 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 at or above 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:

a) is at or above the current value of the Thermal Management Temperature 1 field; and
b) is below the current value of the Thermal Management Temperature 2 field;

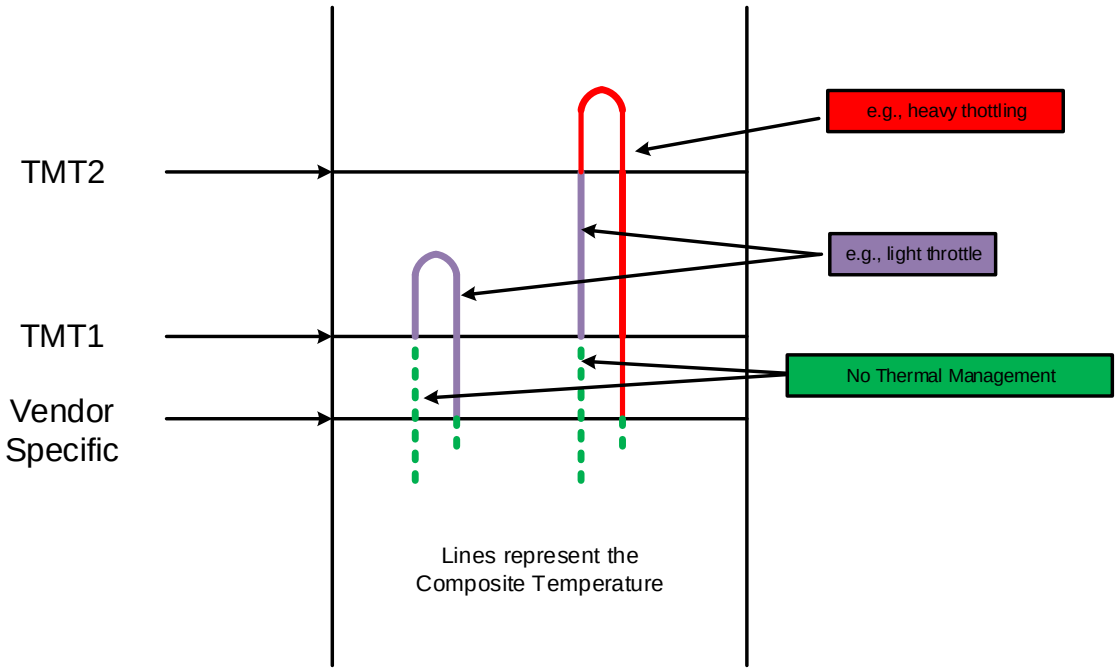
and the Composite Temperature decreases to a value below the current value of the Thermal Management Temperature 1 field, then the controller should return to the active power state that the controller was in prior to going to a lower power active power state or stop performing vendor specific thermal 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 at or above the current value of the Thermal Management Temperature 2 field and the Composite Temperature decreases to 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, 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 264 shows examples of how the Composite Temperature may be effected by this feature.

Figure 264: 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.

8.5 Virtualization Enhancements (Optional)

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.

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 8.5.1).
- Virtual Interrupt Resource (VI Resource): a type of controller resource that manages one interrupt vector (refer to section 8.5.2).

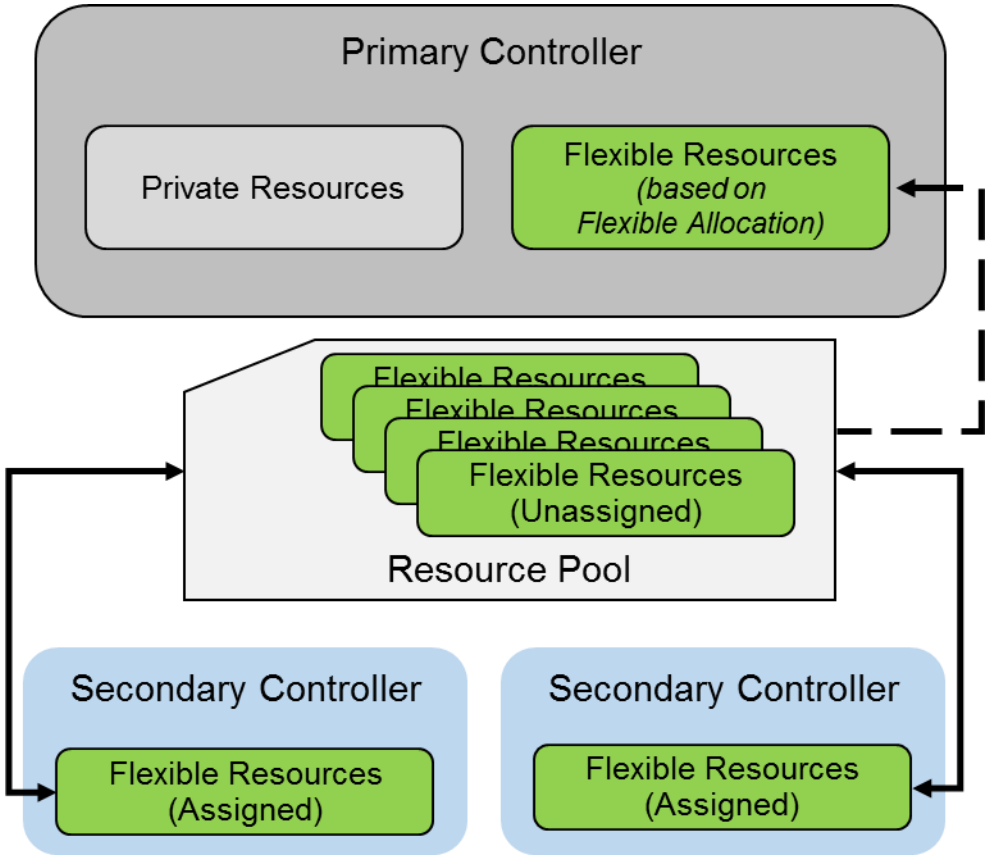
Flexible Resources are controller resources that may be assigned to the primary controller or one of its secondary controllers. The Virtualization Management command is used to provision the Flexible Resources between a primary controller and one of its secondary controller(s). A primary controller's

allocation of Flexible Resources may be modified using the Virtualization Management command and the change takes effect after a Controller Level Reset. A secondary controller only supports having Flexible Resources assigned or removed when it is 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 265 shows the controller resource allocation model for a controller resource type that is assignable as a Flexible Resource.

Figure 265: Controller Resource Allocation



For each controller resource type supported, the Primary Controller Capabilities Structure 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 allocated to the primary controller using the Virtualization Management command;
- 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 in the case of where commands are clearly marked as primary controller only. It is recommended that only primary controllers support privileged actions 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 8.5.3 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;
  - 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 110 (Identify command with CNS value of 14h);
  - The Secondary Controller List defined in Figure 111 (Identify command with CNS value of 15h);
  - The Namespace Management and Namespace Attachment commands;
- One or more secondary controllers;
- 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 8.5.4), 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.

8.5.1 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 5.21.1.7).

If a secondary controller has no VQ Resources assigned to it, then it remains in the Offline state. A secondary controller cannot transition to the Online state until it has VQ Resources for an Admin Queue and one or more I/O Queues assigned to it (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 queues that are Private Resources to ensure there is a minimum of an Admin Queue and one I/O Queue 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 5.21.1.7).

8.5.2 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 it, MSIXCAP.MXC.TS (refer to section 2.4.2) 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 VI Resources assigned to it, then it remains in the Offline state. A secondary controller cannot transition to the Online state until it has a VI Resource for interrupt vector 0 assigned to it. For a secondary controller that supports VI Resources with MSI-X vectors, if no VI Resources are assigned to it, then MSIXCAP.MXC.TS is reserved.

A primary controller that supports VI Resources shall have at least one interrupt that is a Private Resource. As such, the primary controller always has interrupt vector 0 assigned to it. 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 it to any valid interrupt vector, regardless of whether they have the same resource identifier, as long as the CQ and the interrupt vector are attached to the same controller.

8.5.3 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').

- **Offline:** The secondary controller may not be used by a host. CSTS.CFS shall be set to '1'. Controller registers 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.
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 may only be enabled when it is 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 transition to the Offline state implicitly.

Resources shall only be assigned to a secondary controller when it is 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 0h or NumVFs specifies a value that does not enable the associated secondary controller then the secondary controller shall transition to the Offline state implicitly.

8.5.4 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 it 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 1.4.1).

SR-IOV Virtual Functions (VFs) with an NVM Express Class Code (refer to section 2.1.5) 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, as specified in sections 2.1.10, 2.1.11, 2.1.14, and 2.1.15. For a controller that is a PF, SR-IOV Capability register VF BAR2 shall not support Index/Data Pair (refer to section 2.1.12).

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' (refer to section 2.4.3) and MSI-X PBA BIR = '2' (refer to section 2.4.4) are valid for controllers that are VFs.

While the controller registers 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 registers are hidden, but their values are not reset.

8.6 Doorbell Stride for Software Emulation

The doorbell stride, specified in CAP.DSTRD, 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 NVM Express, the expected doorbell stride value is 0h.

8.7 Standard Vendor Specific Command Format

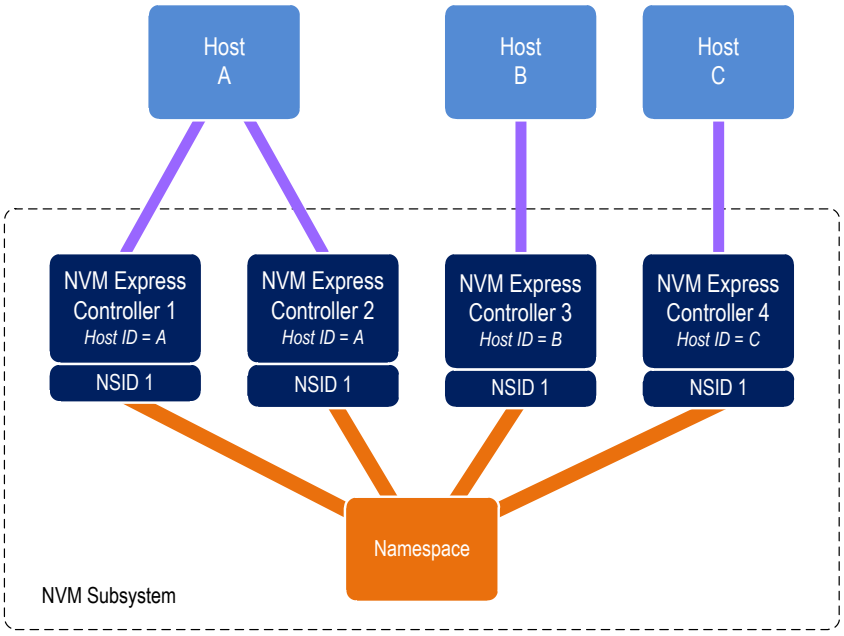
Controllers may support the standard Vendor Specific command format defined in Figure 12. Host storage drivers may use the Number of Dwords fields to ensure that the application is not corrupting physical memory (e.g. overflowing a data buffer). The controller indicates support of this format in the Identify Controller data structure in Figure 109; refer to Admin Vendor Specific Command Configuration and NVM Vendor Specific Command Configuration.

8.8 Reservations (Optional)

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.

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 of Reservation Conflict. Capabilities are provided that allow recovery from a reservation on a namespace held by a failing or uncooperative host.

Figure 266: Example Multi-Host System



A reservation requires an association between a host and a namespace. As shown in Figure 266, 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 266 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 (Host Identifier) with each controller with which it is associated using a Set Features command 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. 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 status 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, Reservation Register command, Reservation Acquire command, and Reservation Release command;
- 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 field of Identify Namespace. For the previous definition of Ignore Existing Key behavior, refer to revision 1.2.1.

8.8.1 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 5.14.1.9.1). Reservation notifications may be masked from generating a reservation log page on a per reservation notification type and per namespace ID basis through the Reservation Notification Mask feature (refer to section 5.21.1.20). A host may use the Asynchronous Event Request command to be notified of the presence of one or more available Reservation Notification log pages (refer to section 5.2).

8.8.2 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 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 it is associated (i.e., that has the same Host Identifier, refer to section 5.21.1.19) to access that namespace as a registrant. Thus, a host need only register on a single controller in order 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 on the namespace with the Reservation Register Action (RREGA) field set 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. It is an error for a host that is already a registrant of a namespace to register with the same namespace using a different registration key value (i.e., the command is aborted with status 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 its existing reservation key by executing a Reservation Register command on the namespace with the RREGA field set to 010b (i.e., Replace Reservation Key), supplying the current reservation key in the Current Reservation Key (CRKEY) field, and the new reservation key in the NRKEY 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 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.

8.8.3 Reservation Types

NVM Express 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 267: 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 267 and the specific behavior for each NVM Express command is shown in Figure 268.

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 or a Set Features command.

Figure 268: 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
NVM Read Command Group: Read Compare Security Receive (Admin)	A	A	C	C	A	A	C	A
NVM Write Command Group: Write Write Uncorrectable Write Zeroes Dataset Management Flush Format NVM (Admin) Namespace Attachment (Admin) Namespace Management (Admin) Sanitize (Admin) Security Send (Admin)	C	C	C	C	C	A	C	A
Reservation Acquire - Acquire	C	C	C	C	C	C	C	C
Reservation Release Reservation Acquire - Preempt Reservation Acquire - Preempt and Abort	C	A	C	A	C	A	C	A
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 status Reservation Conflict Notes: 1. The behavior of a vendor specific command is vendor specific.								

8.8.4 Unregistering

A host that is a registrant of a namespace may unregister with the namespace by executing a Reservation Register command 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 of Reservation Conflict. If the host is not a registrant, then the command is aborted with a status 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.

8.8.5 Acquiring a Reservation

In order for a host to obtain a reservation on a namespace, it shall be a registrant of that namespace. A registrant obtains a reservation by executing a Reservation Acquire command, setting 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 key value does not match, then the command is aborted with status Reservation Conflict. If the host is not a registrant, then the command is aborted with a status 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 status Reservation Conflict. If a reservation holder attempts to obtain a reservation of a different type on a namespace for which it already is the reservation holder, then the command is aborted with status Reservation Conflict. It is not an error if a reservation holder attempts to obtain a reservation of the same type on a namespace for which it already is the reservation holder. A reservation holder may preempt a reservation to change the reservation type.

8.8.6 Releasing a Reservation

Only a reservation holder may release in an orderly manner a reservation held on a namespace. A host releases a reservation by executing a Reservation Release command, setting the Reservation Release Action (RRELA) field to 000b (i.e., Release), setting the Reservation Type (RTYPE) field to the type of reservation being released, 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 host to register with the namespace. If the key value doesn't match, then the command is aborted with status Reservation Conflict. If the RTYPE field does not match the type of the current reservation, then the command completes with status 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.

8.8.7 Preempting a Reservation or Registration

A host that is a registrant may preempt a reservation and/or registration by executing a Reservation Acquire command, 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 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 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. If the PRKEY field value matches the reservation key of the current reservation holder, then the following occur as an atomic operation: the reservation holder is 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 as the reservation key holder. If the PRKEY field value does not match that of the current reservation holder and is not equal to zero, 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 zero, then the command is aborted with status 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. If the PRKEY field value is zero, 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 for the host of the type specified by the Reservation Type (RTYPE) field in the command. 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.

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 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 target the namespace specified in the Namespace Identifier field (CDW1.NSID of the Reservation Acquire command) (refer to section 4.11 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 Admin command, abort 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 it can no longer 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.

8.8.8 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 executing a Reservation Release command, setting the Reservation Release Action (RRELA) field to 001b (i.e., Clear), and 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 status Reservation Conflict. If the host is not a registrant, then the command is aborted with a status of Reservation Conflict. When a reservation is cleared the following occur as an atomic operation: any reservation held on the namespace 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.

8.8.9 Reporting Reservation Status

A host may determine the current reservation status associated with a namespace by executing a Reservation Report command.

8.9 Host Memory Buffer (Optional)

The Host Memory Buffer 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 5.21.1.13.

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 it is no longer using the ranges, 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 Replay Protected Memory Block (Optional)

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 register 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. 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 270 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 269 defines the RPMB Device Configuration Block data structure – the non-volatile contents stored within the controller for RPMB target 0.

Figure 269: 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 to 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 8.13.3. Bits 7 to 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.
511:02		Reserved

Each RPMB Data Frame is 256 bytes in size plus the size of the Data field, and is organized as shown in Figure 273. RPMB uses a sector size of 512 bytes. The RPMB sector size is independent and not related to the logical block size used for the namespace(s).

Figure 270: 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

Response Message Types		Description	Requires Data	RPMB Frame Length (bytes)
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 271 indicates whether an RPMB request was successful or not.

Figure 271: RPMB Operation Result

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>15:08</td><td>Reserved</td></tr>
<tr><td>07</td><td>Write Counter Status: Indicates if the Write Counter has expired (i.e., reached its maximum value). A value of one indicates that the Write Counter has expired. A value of zero indicates a valid Write Counter.</td></tr>
<tr><td>06:00</td><td>Operation Status: Indicates the operation status. Valid operation status values are listed below.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>00h</td><td>Operation successful</td></tr>
<tr><td>01h</td><td>General failure</td></tr>
<tr><td>02h</td><td>Authentication failure (MAC comparison not matching, MAC calculation failure)</td></tr>
<tr><td>03h</td><td>Counter failure (counters not matching in comparison, counter incrementing failure)</td></tr>
<tr><td>04h</td><td>Address failure (address out of range, wrong address alignment)</td></tr>
<tr><td>05h</td><td>Write failure (data/counter/result write failure)</td></tr>
<tr><td>06h</td><td>Read failure (data/counter/result read failure)</td></tr>
<tr><td>07h</td><td>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.</td></tr>
<tr><td>08h</td><td>Invalid RPMB Device Configuration Block – this may be used when the target is not 0.</td></tr>
</table></td></tr>
</table>

Figure 272 defines the non-volatile contents stored within the controller for each RPMB target.

Figure 272: 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)	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 register 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 (128KB minimum, 32MB maximum)	Data which may only be read and written 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 273. RPMB uses a sector size of 512 bytes. The RPMB sector size is independent and not related to the logical block size used for the namespace(s).

Figure 273: 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*).
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).
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 271. Note: The Result field is not needed for Requests.
255:254	Request/Response Message	Defined in Figure 270.
(*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 273. 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.

8.10.1 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 then the message authentication code (MAC) is calculated using HMAC SHA-256 as defined in [HMAC-SHA]. 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.

8.10.2 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.

8.10.2.1 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 274: 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	0000 0000h	
	247:244	Address	0000 0000h	
	251:248	Sector Count	0000 0000h	
	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	0000 0000h	
	247:244	Address	0000 0000h	
	251:248	Sector Count	0000 0000h	
	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 to access*	
	239:224	Nonce	0…00h	
	243:240	Write Counter	0000 0000h	
	247:244	Address	0000 0000h	
	251:248	Sector Count	0000 0000h	
	253:252	Result	*Result Code*	
	255:254	Request/Response	0100h *(Response)*	

8.10.2.2 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 275: 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	0000 0000h	
	247:244	Address	0000 0000h	
	251:248	Sector Count	0000 0000h	
	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	0000 0000h	
	251:248	Sector Count	0000 0000h	
	253:252	Result	*Result Code*	
	255:254	Request/Response	0200h *(Response)*	

8.10.2.3 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, it first checks whether the Write Counter has expired. If the Write Counter has expired then the controller sets the result to 0085h (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 8.10.1) 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 03h (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 register 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.
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 276: 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	0000 0000h	
	247:244	Address	0000 0000h	
	251:248	Sector Count	0000 0000h	
	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	0000 0000h	
	253:252	Result	*Result Code*	
	255:254	Request/Response	0300h *(Response)*	

8.10.2.4 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, it 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, it 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 277: 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	0000 0000h	
	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*	

8.10.3 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 278). All other fields are cleared to 0h.

If the Write Counter has expired then the controller sets the result to 0085h (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 08h (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, it 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 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 8.10.4).

Figure 278: 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	0000 0000h	
	251:248	Sector Count	0000 0001h	
	253:252	Result	0000h	
	255:254	Request/Response	0006h *(Request)*	
	767:256	Data	*RPMB Device Configuration Block data structure*	
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	0000 0000h	
	251:248	Sector Count	0000 0000h	
	253:252	Result	*Result Code*	
	255:254	Request/Response	0600h *(Response)*	

8.10.4 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, it 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 269), 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 278.

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 279: 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	0000 0000h	
	247:244	Address	0000 0000h	
	251:248	Sector Count	0000 0001h	
	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	0000 0000h	
	251:248	Sector Count	0000 0001h	
	253:252	Result	*Result Code*	
	255:254	Request/Response	0700h *(Response)*	
	767:256	Data	*RPMB Device Configuration Block data structure*	

8.11 Device Self-test Operations (Optional)

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 in to 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 5.14.1.6) 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 8.11.1)

b) extended device self-test operation (refer to section 8.11.2).

Figure 280 is an informative example of a device self-test operation with the associated segments and tests performed in each segment.

Figure 280: 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
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	

8.11.1 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 5.14.1.6).

A short device self-test operation:

a) shall be aborted by any Controller Level Reset;
b) shall be aborted by a Format NVM, if the Namespace Identifier field specified in the Format NVM command is the same as the Device Self-test command that invoked the device self-test operation;
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.

8.11.2 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 5.14.1.6).

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, if the Namespace Identifier field specified in the Format NVM command is the same as Device Self-test command the invoked the device self-test operation;
b) shall be aborted if a Device Self-test command with the Self-Test Code field set to Fh is processed; and
c) may be aborted if the specified namespace is removed from the namespace inventory.

8.12 Namespace Management (Optional)

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. Namespace management is intended for use during manufacturing or by a system administrator.

When a namespace is detached from a controller or deleted it becomes an inactive namespace on that controller. Previously submitted but uncompleted or subsequently submitted commands to the affected namespace are handled by the controller as if they were issued to an inactive namespace.

The size of a namespace is based on the number of logical blocks 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.

The total and unallocated NVM capacity for the NVM subsystem is reported in the Identify Controller data structure. For each namespace, the NVM capacity used for that namespace is reported in the Identify Namespace data structure. 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. For example, if host software requests a namespace of 32 logical blocks with a logical block size of 4KB for a total size of 128KB and the allocation unit for the implementation is 1MB then the NVM capacity consumed may be rounded up to 1MB. The NVM capacity fields may not correspond to the logical block size multiplied by the total number of logical blocks.

To create a namespace, host software performs the following actions:

1. Host software requests the Identify Namespace data structure that specifies common namespace capabilities (Identify with a setting of CDW1.NSID set to FFFFFFFFh and CNS set to 0h).
2. Host software creates the data structure defined in Figure 129. Host software sets the host software specified fields defined in Figure 126 to the desired values (taking into account the common namespace capabilities).
3. 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.
4. Host software requests the Identify Namespace data structure 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 new namespace to one or more controllers.
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 namespace from one or more controllers.
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.
3. If Namespace Attribute Notices are enabled, any controller(s) that was attached to the namespace reports a Namespace Attribute Changed asynchronous event to the host.

8.13 Boot Partitions (Optional)

Boot Partitions provide an optional area of NVM storage that may be read 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 storage medium (e.g., SPI flash). Refer to section 8.13.1 for the procedure to read the contents of a Boot Partition.

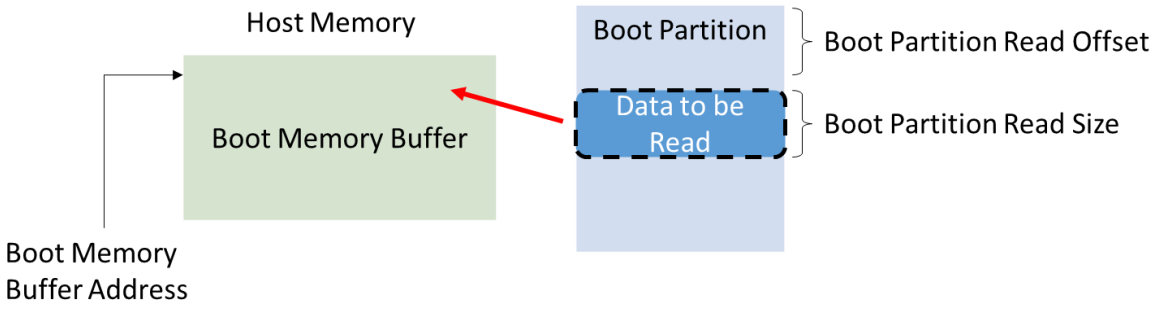
An NVMe controller that supports Boot Partitions has two Boot Partitions of equal size using Boot Partition identifiers 0h and 1h. The two Boot Partitions allow the host to update one and verify the contents before marking the Boot Partition active.

The contents of Boot Partitions are only modified using the Firmware Image Download and Firmware Commit commands (refer to section 8.13.2) and may be secured using Replay Protected Memory Block to prevent unauthorized modifications (refer to section 8.13.3).

8.13.1 Reading from a Boot Partition

A Boot Partition is a continuous block of data as shown in Figure 281, that the host may read.

Figure 281: Boot Partition Overview



To read the contents of a Boot Partition, 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 the PCI Express registers (described in section 2), 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) register. 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 field (BPINFO.BRS) while transferring the Boot Partition contents to indicate a Boot Partition read operation is in progress.
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 2h, the requested Boot Partition data has been transferred to the Boot Partition Memory Buffer. If BPINFO.BRS is set to 3h, 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.

8.13.2 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 the 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 8.13.3).
3. The host submits a Firmware Commit command 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 8.13.1). Host software updates the active Boot Partition ID by issuing a Firmware Commit command with a Commit Action of 111b.
6. The host locks Boot Partition access to prevent further modification (refer to section 8.13.3).

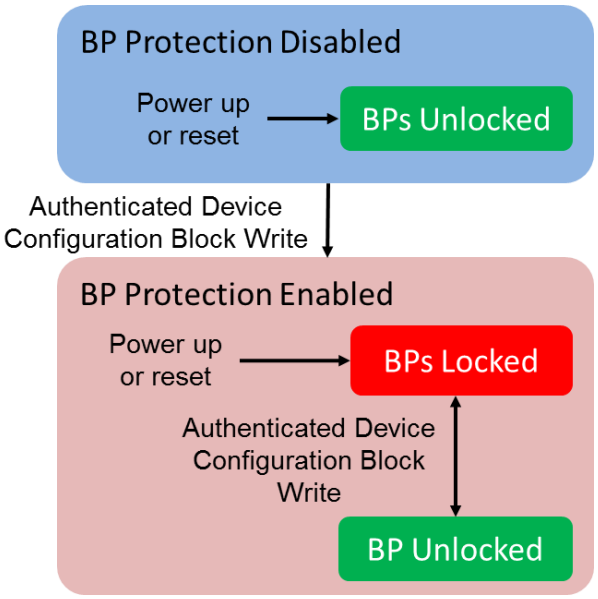
If an internal error, reset, or power loss condition occurs while committing the downloaded image to a Boot Partition, the contents of the Boot Partition may contain the old contents, new contents, or a mixture of both. Host software should verify the contents of a Boot Partition before marking it active 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.

8.13.3 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 8.10). Figure 282 shows an overview of Boot Partition Protection.

Figure 282: 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 8.10). 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 may only be modified 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.

8.14 Telemetry (Optional)

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 5.14.1.7 and 5.14.1.8). 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 in the Log Page Attributes field in the Identify Controller data structure.

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 making it difficult to determine the root cause. Given this, flexibility in size is desired 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 in order to identify the source of the problem. For rare issues that are not root caused by a small or medium sized telemetry data collection, a third phase may be employed to collect the largest and most complete payload to diagnose the issue.

The telemetry data is returned in Telemetry Data Blocks. Each Telemetry Data Block is 512 bytes in size. A set of Telemetry Data Blocks forms a Telemetry Data Area. Each Telemetry Data Area represents the controller's internal state when the telemetry data was captured. There are three Telemetry Data Areas defined in the host-initiated and controller-initiated logs, all starting at Telemetry Data Block 1. Each 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 (first phase), Telemetry Data Area 2 is intended to have a medium size payload (second phase), and Telemetry Data Area 3 is intended to have a large size payload (third phase). The size of each area is vendor specific and may change on each data collection. When possible, the host should retrieve the payload for all three 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 prepares host memory buffer(s) to store telemetry data if needed.
3. To receive notification that controller-initiated data is available, the host enables Telemetry Log Notices using the Asynchronous Event Configuration feature.
4. 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 in the host-initiated or controller-initiated log. If possible, the host should collect Telemetry Data Area 1, 2, and 3. The host reads the log in 512 byte Telemetry Data Block units. As part of the last read for a controller-initiated log, the host clears the Retain Asynchronous Event bit to '0'.
    b. If it is a controller-initiated log, the host re-reads the header of the log page and ensures that the Telemetry Controller-Initiated Data Generation Number matches the original value read. If it does not match, then the data captured is not consistent and needs to be re-read.
    c. 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’. 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 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.

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 it 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.

8.14.1 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.

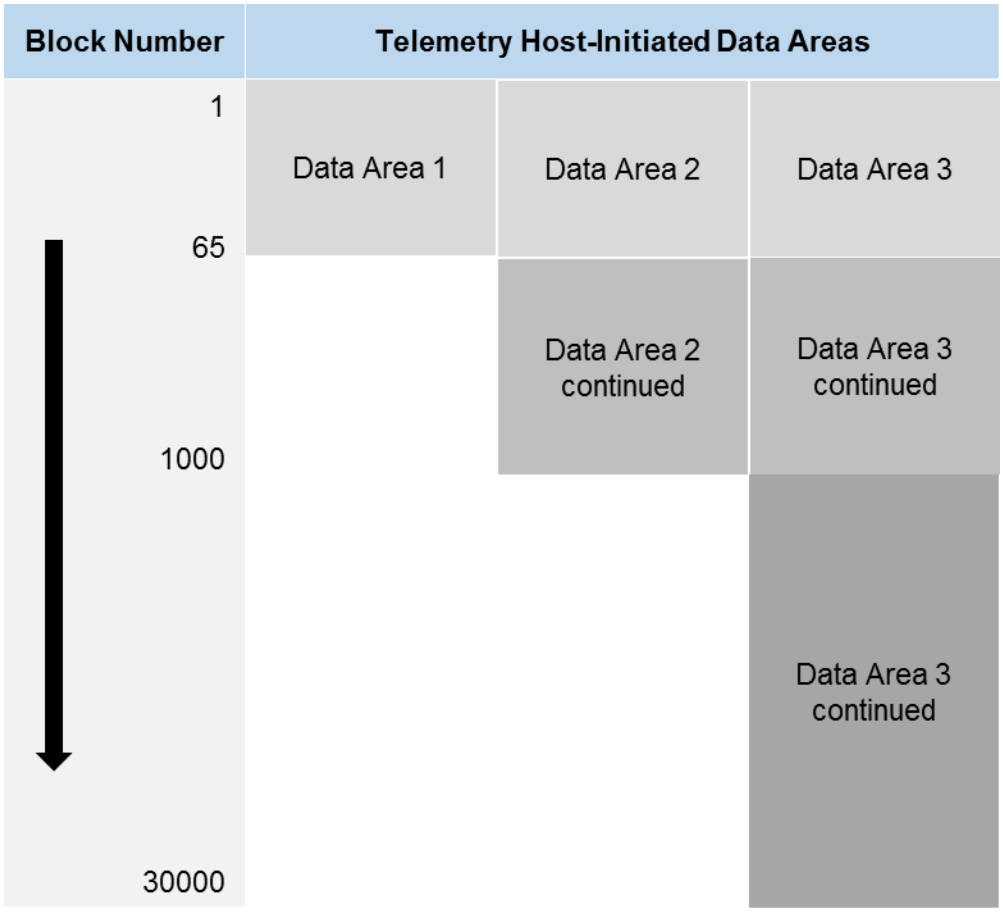
If a Telemetry Host-Initiated log page has no data for collection then the following fields are all cleared to 0h:

- Telemetry Host-Initiated Data Area 1 Last Block = 0,
- Telemetry Host-Initiated Data Area 2 Last Block = 0,
- 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. As an example, the following values correspond to the layout shown in Figure 283:

- Telemetry Host-Initiated Data Area 1 Last Block = 65,
- Telemetry Host-Initiated Data Area 2 Last Block = 1000,
- Telemetry Host-Initiated Data Area 3 Last Block = 30000.

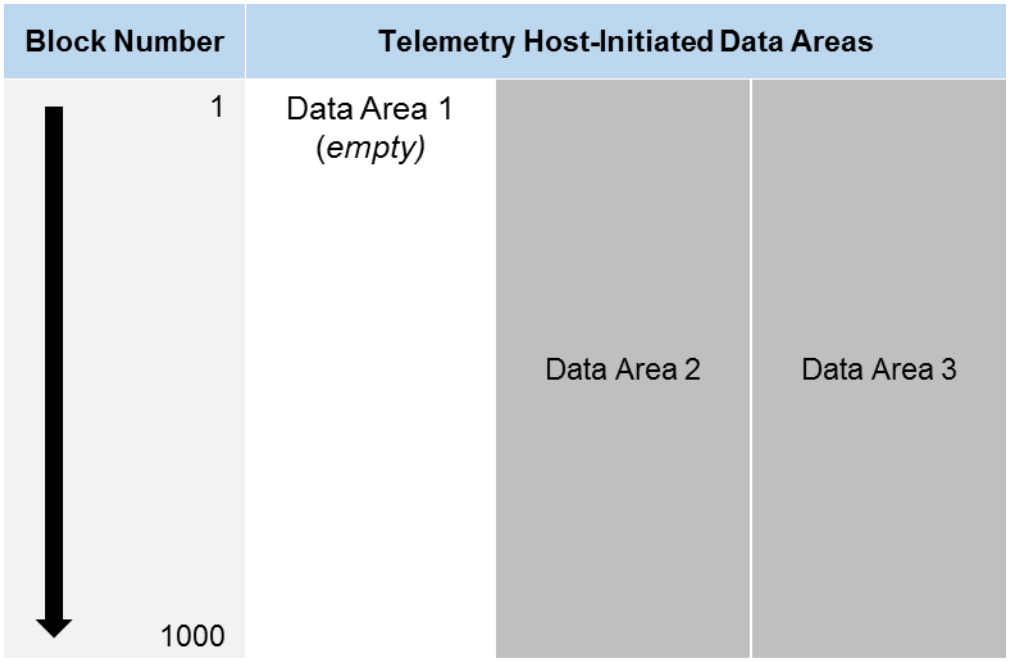
Figure 283: Telemetry Log Example – All Data Areas Populated



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. As an example, the following values correspond to the layout shown in Figure 284:

- Telemetry Host-Initiated Data Area 1 Last Block = 0,
- Telemetry Host-Initiated Data Area 2 Last Block = 1000,
- Telemetry Host-Initiated Data Area 3 Last Block = 1000.

Figure 284: Telemetry Log Example – Data Area 2 Populated



8.15 Sanitize Operations (Optional)

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 such as NVMe-MI.

The scope of a sanitize operation is all locations in the NVM subsystem that are able to contain user data, including caches and unallocated or deallocated areas of the media. 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). Once a sanitize operation is started, it cannot be aborted and continues after a Controller Level Reset including across power cycles.

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 the Sanitize Operation Completed asynchronous event (if an Asynchronous Event Request Command is outstanding).

The Sanitize Capabilities field of the Identify Controller data structure indicates the sanitize operation types supported.

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.
- 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 285 defines the data pattern or patterns that are written.

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 285: Sanitize Operations – Overwrite Mechanism

OIPBP[1]	**Overwrite Pass Count**[1]	**Overwrite Pass Number**	**Logical Block Data and Non-PI Metadata**[2]	**Protection Information**[3]
'0'	All	All	Overwrite Pattern[1]	FFFFFFFF_FFFFFFFFh
'1'	Even	First	Inversion of Overwrite Pattern[1]	00000000_00000000h
		Subsequent	Inversion of Overwrite Pattern[1] from previous pass (i.e., each bit XORed with '1')	
'1'	Odd	First	Overwrite Pattern[1]	FFFFFFFF_FFFFFFFFh
		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 metadata other than Protection Infomation is present.
3. 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, to the desired values. 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 a Sanitize command is completed with any status 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. While a sanitize operation is in progress, all controllers shall abort any commands not listed in Figure 287 with a status of Sanitize In Progress (refer to section 8.15.1).

The user data values that result from a successful sanitize operation are specified in Figure 286. If the controller deallocates user data after successful completion of a sanitize operation, then values read from deallocated logical blocks are described in section 6.7.1.1. The host may specify that sanitized logical blocks not be deallocated by setting the No Deallocate After Sanitize bit to '1' in the Sanitize command.

Figure 286: Sanitize Operations – User Data Values

Sanitize Operation	Logical Blocks	Non-PI Metadata[1]	Protection Information[2]
Block Erase	Vendor specific value	Vendor specific value	Vendor specific value
Crypto Erase	Indeterminate	Indeterminate	Indeterminate
Overwrite	Refer to Figure 285	Refer to Figure 285	Refer to Figure 285

NOTES:
1. If metadata other than Protection Information is present.
2. If Protection Information is present within the metadata.

The Sanitize Status log page (refer to section 5.14.1.9.2) 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. 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.
- 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).
- 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).

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.
  - The Asynchronous Event Type field is set to 6h (i.e., I/O Command Set specific status).
- All controllers in the NVM subsystem may resume any power management that was suspended when the sanitize operation started.

The host should read the Sanitize Status log page upon completion of a sanitize operation (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 of Sanitize Failed (refer to section 8.15.1) 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 and enable the event by default;
- 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; and
- Indicate support for all supported sanitize operation types in the Sanitize Capabilities field in the Identify Controller data structure.

8.15.1 Command 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 287 subject to the additional restrictions stated in that figure;
- All I/O Commands shall be aborted with a status of Sanitize In Progress; and
- Any command or command option that is not explicitly permitted in Figure 287 shall be aborted with a status of Sanitize in Progress if fetched by any controller in the NVM subsystem.

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 287 subject to the additional restrictions noted in that figure;
- All I/O Commands are shall be aborted with a status of Sanitize Failed;
- The Sanitize command is permitted with action restrictions (refer to section 5.24); and
- Aside from the Sanitize command, any other command or command option that is not explicitly permitted in Figure 287 shall be aborted with a status of Sanitize Failed if fetched by any controller in the NVM subsystem.

Figure 287: Sanitize Operations – Admin Commands Allowed

<table>
<tr><th>Admin Command</th><th>Additional Restrictions</th></tr>
<tr><td>Abort</td><td></td></tr>
<tr><td>Asynchronous Event Request</td><td></td></tr>
<tr><td>Create I/O Completion Queue</td><td></td></tr>
<tr><td>Create I/O Submission Queue</td><td></td></tr>
<tr><td>Delete I/O Completion Queue</td><td></td></tr>
<tr><td>Delete I/O Submission Queue</td><td></td></tr>
<tr><td>Get Features</td><td></td></tr>
<tr><td>Get Log Page</td><td>The log pages allowed are listed below.

<table><tr><th>Log Pages</th><th>Additional Restrictions</th></tr><tr><td>Error Information</td><td>Return zeros 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>Sanitize Status</td><td></td></tr></table></td></tr>
<tr><td>Identify</td><td></td></tr>
<tr><td>Keep Alive</td><td></td></tr>
<tr><td>Set Features</td><td></td></tr>
<tr><td>Opcode 7Fh</td><td>The Fabric Commands allowed are listed below. Refer to the NVMe over Fabrics specification.

<table><tr><th>Fabrics Commands</th><th>Additional Restrictions</th></tr><tr><td>Property Set</td><td></td></tr><tr><td>Connect</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>Commands are allowed that do not affect or retrieve user data.</td></tr>
</table>

9 Directives

Directives is a mechanism to enable host and NVM subsystem or controller information exchange. The Directive Receive command is used to transfer data related to a specific Directive Type from the controller to the host. The Directive Send command 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.

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 288) are the Identify Directive (refer to section 9.2), and the Streams Directive (refer to section 9.3). 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 288: Directive Types

Directive	Directive Type Value	Definition	I/O Command Directive
Identify	00h	Section 9.2	No
Streams	01h	Section 9.3	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 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.

9.1 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 upper four bits are assumed to be zero. A Directive Type of 00h in an I/O command specifies that the I/O command is not using Directives.

The only I/O command that supports use of directives in this version of this specification is the Write command.

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 289).

Figure 289: 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 – 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 of Invalid Field in Command.

For the Streams Directive (i.e., DTYPE field set to 01h), if the DSPEC field is cleared to 0000h in a Write command, then that Write command shall be processed as a normal write operation (i.e., as if DTYPE field is cleared to 00h).

9.2 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 290.

Figure 290: Identify Directive – Directive Operations

Directive Command	Directive Operation Name	Directive Operation Value	Definition
Directive Receive	Return Parameters	01h	Section 9.2.1.1
	Reserved	All others	
Directive Send	Enable Directive	01h	Section 9.2.2.1
	Reserved	All others	

9.2.1 Directive Receive

This section defines operations used with the Directive Receive command for the Identify Directive.

9.2.1.1 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 291. If an NSID value of FFFFFFFFh is specified, then then the controller shall abort the command with a status of Invalid Field in Command. The DSPEC field in command Dword 11 is not used for this operation.

Figure 291: Identify Directive – Return Parameters Data Structure

Bytes	Bit	Description
		Directives Supported
31:00	255:02	Reserved
	01	**Streams Directive:** This bit is set to 1b if the Streams Directive is supported. This bit is cleared to 0b if the Streams Directive is not supported.
	00	**Identify Directive:** This bit shall be set to 1b to indicate that the Identify Directive is supported.
		Directives Enabled
63:32	255:02	Reserved
	01	**Streams Directive:** This bit is set to 1b if the Streams Directive is enabled. This bit is cleared to 0b if the Streams Directive is not enabled.
	00	**Identify Directive:** This bit shall be set to 1b to indicate that the Identify Directive is enabled.
4095:64	n/a	Reserved

9.2.2 Directive Send

This section defines operations used with the Directive Send command for the Identify Directive.

9.2.2.1 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 for the entire NVM subsystem. On any other type of 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, 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 292: Enable Directive – Command Dword 12

Bit	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 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 field is set to '1' for a Directive that is not supported, then a status of Invalid Field in Command shall be returned.

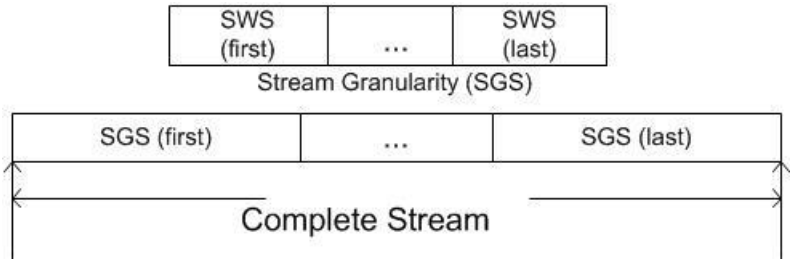
9.3 Streams (Directive Type 01h, Optional)

The Streams Directive enables the host to indicate (i.e., by using the stream identifier) to the controller that the specified 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 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 293.

Figure 293: Directive Streams – Stream Alignment and Granularity



If the host issues a Dataset Management command to deallocate logical blocks that are associated with a stream, it 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. Stream resources may be allocated for the exclusive use of a specified namespace associated with a particular Host Identifier using the Allocate Resources operation. Stream resources that are not allocated for the exclusive use of any namespace are available NVM subsystem stream resources as reported in NVM Subsystem Streams Available (NSSA) and may be used by any namespace that has the Streams Directive enabled and has not been allocated exclusive stream resources in response to an Allocate Resources operation. As stream resources are allocated for the exclusive use of a specified namespace, the available NVM subsystem stream resources reported in the NSSA field are reduced.

The Directive operations that shall be supported if the Streams Directive is supported are listed in Figure 294. 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 294: Streams – Directive Operations

Directive Command	Directive Operation Name	Directive Operation Value	Definition
Directive Receive	Return Parameters	01h	Section 9.3.1.1
	Get Status	02h	Section 9.3.1.2
	Allocate Resources	03h	Section 9.3.1.3
	Reserved	All others	
Directive Send	Release Identifier	01h	Section 9.3.2.1
	Release Resources	02h	Section 9.3.2.2
	Reserved	All others	

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 be accessing a namespace through multiple controllers in the NVM subsystem. The controllers in an NVM subsystem distinguish if the stream identifier has the same meaning for a particular namespace by the Host Identifier. If more than one Host Identifier has the same non-zero value, then that value represents a single host that is accessing the namespace through multiple controllers and the stream identifier is used across controllers to access the same stream on the namespace. If a Host Identifier is zero or has a unique value, then that value represents a unique host that is accessing the namespace and the stream identifier does not have the same meaning for a particular namespace.

The controller(s) recognized by the NVM subsystem as being associated with a specific host 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 of 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 of 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 streams allocated, if the host submits a write 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 of 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, then all stream resources and stream identifiers are released for the affected namespace. If the host issues a Format NVM command, or deletes a namespace, then all stream identifiers for all open streams for affected namespaces are released.

Streams Directive defines the command specific status values specified in Figure 295.

Figure 295: 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.

9.3.1 Directive Receive

This section defines operations used with the Directive Receive command for the Streams Directive.

9.3.1.1 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 296. 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 zero.

Figure 296: Streams Directive– Return Parameters Data Structure

Bytes	Description
	NVM Subsystem Specific Fields
1:0	**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.
3:2	**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 of any specific namespace. This field returns the same value independent of specified namespace.
5:4	**NVM Subsystem Streams Open (NSSO)**: This field indicates the number of open streams in the NVM subsystem that are not associated with a namespace with allocated stream resources. This field returns the same value independent of specified namespace.
15:6	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 of logical blocks for this namespace. 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 LBA format. If the NSID value is set to FFFFFFFFh then this field may be cleared to 0h if a single logical block size cannot be indicated.
21:20	**Stream Granularity Size (SGS):** This field indicates the stream granularity size for this namespace in Stream Write Size (SWS) units. If the NSID value is set to FFFFFFFFh then this field may be cleared to 0h.
23:22	**Namespace Streams Allocated (NSA):** This field indicates the number of stream resources allocated for exclusive use with 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 zero, 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 namespace. This field is cleared to zero if no stream resources are currently allocated to this namespace.
31:26	Reserved

9.3.1.2 Get Status (Directive Operation 02h)

The Get Status operation returns information about the status of currently open streams for the specified namespace. The DSPEC field in command Dword 11 is not used for this operation. If an NSID value of FFFFFFFFh is specified then the controller shall return information about the status of currently open streams for the NVM subsystem that are not associated with any namespace that has allocated stream resources for its exclusive use.

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 297. All fields are specific to the namespace specified.

Figure 297: 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 65535:** This field specifies the stream identifier of the 65535 open stream.

9.3.1.3 Allocate Resources (Directive Operation 03h)

The Allocate Resources operation indicates the number of streams that the host requests for the exclusive use of the specified namespace. 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 of this namespace, the controller shall:

- return a status value of Stream Resource Allocation Failed; or
- if NVM subsystem stream resources are available, then set NSA to 0000h in the completion queue entry to indicate that the host may use stream resources from the NVM subsystem for this namespace.

If the namespace already has been allocated streams resources for its exclusive use, the controller shall return a status value of Invalid Field in Command. To allocate additional streams resources, the host should release resources and then request a complete set of resources.

No data transfer occurs.

Figure 298: Allocate Resources – Command Dword 12

Bit	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 299: Allocate Resources – Dword 0 of command completion queue entry

Bit	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.

9.3.2 Directive Send

This section defines operations used with the Directive Send command for the Streams Directive.

9.3.2.1 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 the stream identifier in a

future operation then it is referring to a different stream. If the specified identifier does not correspond to an open stream for the specified namespace, then the command completes successfully. If there are stream resources allocated for the specified namespace, then the 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 specified namespace, then the stream resources are returned to the NVM subsystem stream resources for future use by a namespace without allocated stream resources. If an NSID value of FFFFFFFFh is specified, then the controller shall abort the command with a status of Invalid Field in Command.

No data transfer occurs.

9.3.2.2 Release Resources (Directive Operation 02h)

The Release Resources operation is used to release all streams resources allocated for the namespace attached to all controllers associated with the same non-zero Host Identifier of the controller that processed the operation. On successful completion of this command, the allocated stream resources are cleared to 0h for the specified namespace. If this command is issued when no streams resources are allocated for the namespace, the command shall complete successfully.

No data transfer occurs.

10 Error Reporting and Recovery

10.1 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.

10.2 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. If the write size is less than or equal to the Atomic Write Unit Power Fail size, then subsequent reads for the associated logical blocks shall return data from the previous successful write operation. If the write size is larger than the Atomic Write Unit Power Fail size, then subsequent reads for the associated logical blocks may return data from the previous successful write operation or this failed write operation.

Based on the value of the Limited Retry bit, the controller may apply all available error recovery means to complete the command.

10.3 Memory Error Handling

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 given a pointer to the host controller that does not exist in memory. When this occurs, the host controller aborts the command with a Data Transfer Error status code.

10.4 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 (if multiple error conditions exist, the lowest numerical value is returned). 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.

10.5 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 or I/O Completion Queues, then the controller may set the Controller Fatal Status (CSTS.CFS) field to '1'. This indicates to host software that a serious error condition has occurred. When this condition occurs, host software should 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) field to determine if a more serious error has occurred.