NVM Express®

NVM Command Set Specification

Revision 1.1

August 5th, 2024

Please send comments to info@nvmexpress.org

NVM Express® NVM Command Set Specification, Revision 1.1 is available for download at http://nvmexpress.org. The NVM Express NVM Command Set Specification, Revision 1.1 incorporates the NVM Express® NVM Command Set Specification, Revision 1.0, ratified on June 3, 2021, ECN 001, ECN102, ECN 104, ECN105, ECN106, ECN108, ECN109, ECN110, ECN111, ECN113, ECN115, ECN116, ECN118, ECN119, ECN120, ECN122, TP 4034a, TP4055, TP 4074a, TP4077, TP 4090, TP4095a, TP 4097a, TP4098a, TP 4099, TP4115, TP4116, TP4130a, TP4135, TP4136, TP4140, TP4141a, TP4146b, TP4148, TP4150, TP4152, TP4159, TP4160, TP4162a, TP4165, TP4171, TP4175, TP4186, TP6036, and TP8012 (refer to https://nvmexpress.org/changes-in-nvme-revision-2-1 for details).

SPECIFICATION DISCLAIMER

LEGAL NOTICE:

© Copyright 2008 to 2024 NVM Express, Inc. ALL RIGHTS RESERVED.

This NVM Express NVM Command Set Specification, Revision 1.1 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 NVM Command Set Specification, Revision 1.1 subject, however, to the Member's continued compliance with the Company's Intellectual Property Policy and Bylaws and the Member's Participation Agreement.

NOTICE TO NON-MEMBERS OF NVM EXPRESS, INC.: If you are not a Member of NVM Express, Inc. and you have obtained a copy of this document, you only have a right to review this document or make reference to or cite this document. Any such references or citations to this document must acknowledge NVM Express, Inc. copyright ownership of this document. The proper copyright citation or reference is as follows: **"© 2008 to 2024 NVM Express, Inc. ALL RIGHTS RESERVED."** When making any such citations or references to this document you are not permitted to revise, alter, modify, make any derivatives of, or otherwise amend the referenced portion of this document in any way without the prior express written permission of NVM Express, Inc. Nothing contained in this document shall be deemed as granting you any kind of license to implement or use this document or the specification described therein, or any of its contents, either expressly or impliedly, or to any intellectual property owned or controlled by NVM Express, Inc., including, without limitation, any trademarks of NVM Express, Inc.

LEGAL DISCLAIMER:

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

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

The NVM Express® design mark is a registered trademark of NVM Express, Inc.
PCIe® is a registered trademark of PCI-SIG.
SNIA® is a registered trademark of the Storage Networking Industry Association (SNIA), USA.

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

Table of Contents

Table of Figures

1 Introduction

1.1 Overview

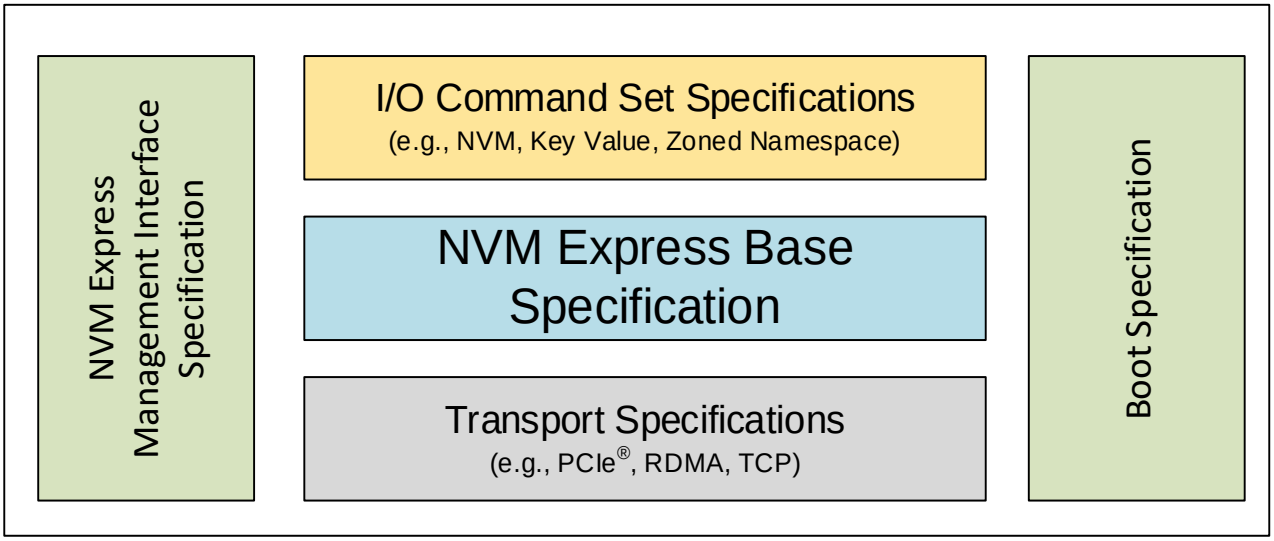
NVM Express® (NVMe®) Base Specification defines an interface for host software to communicate with a non-volatile memory subsystem (NVM subsystem) over a variety of memory-based transports and message-based transports.

This document defines a specific NVMe I/O Command Set, the NVM Command Set, which extends the NVM Express Base Specification.

1.2 Scope

Figure 1 shows the relationship of the NVM Express® NVM Command Set Specification to other specifications within the NVMe Family of Specifications.

Figure 1: NVMe Family of Specifications



This specification supplements the NVM Express Base Specification. This specification defines additional data structures, features, log pages, commands, and status values. This specification also defines extensions to existing data structures, features, log pages, commands, and status values. This specification defines requirements and behaviors that are specific to the NVM Command Set. Functionality that is applicable generally to NVMe or that is applicable across multiple I/O Command Sets is defined in the NVM Express Base Specification.

If a conflict arises among requirements defined in different specifications, then a lower-numbered specification in the following list shall take precedence over a higher-numbered specification:

1. Non-NVMe specifications
2. NVM Express Base Specification
3. NVMe transport specifications
4. NVMe I/O command set specifications
5. NVM Express Management Interface Specification
6. NVM Express Boot Specification

1.3 Conventions

This specification conforms to the Conventions section, Keywords section, and Byte, Word, and Dword Relationships section of the NVM Express Base Specification.

1.4 Definitions

1.4.1 Definitions from the NVM Express Base Specification

This specification uses the definitions in the NVM Express Base Specification.

1.4.2 Definitions in the NVM Express Base Specification specified in the NVM Command set

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

1.4.2.1 Endurance Group Host Read Command

The Compare command, Copy command, Read command, and Verify command.

1.4.2.2 Format Index

A value used to index into the LBA Formats list (refer to Figure 114) and the Extended LBA Formats list (refer to Figure 118).

1.4.2.3 Identify Controller data structures

All controller data structures that are able to be retrieved via the Identify command for the NVM Command Set:

- the Identify Controller data structure (refer to the NVM Express Base Specification and section 4.1.5.2); and
- the I/O Command Set specific Identify Controller data structure for the NVM Command Set (refer to section 4.1.5.4).

1.4.2.4 Identify Namespace data structures

All namespace data structures that are able to be retrieved via the Identify command for the NVM Command Set:

- the I/O Command Set Independent Identify Namespace data structure (refer to the NVM Express Base Specification);
- the Identify Namespace data structure (refer to section 4.1.5.1); and
- the I/O Command Set specific Identify Namespace data structure for the NVM Command Set (refer to section 4.1.5.3).

1.4.2.5 logical block data size

The size in bytes of a logical block, excluding metadata, if any. The size is calculated using the following formula:

2 ^ DataExponent

Where:

- DataExponent is the value in the LBA Data Size field in the NVM Command Set specific LBA Format data structure (refer to Figure 116).

1.4.2.6 logical block size

The size in bytes of a logical block, including metadata size. The size is calculated using the following formula:

Logical block data size + MetadataBytes

where:

- logical block data size is defined in section 1.4.2.5.
- MetadataBytes is the value in the Metadata Size field in the NVM Command Set specific LBA Format data structure (refer to Figure 116).

1.4.2.7 SMART Data Units Read Command

The Compare command, Read command, and Verify command.

1.4.2.8 SMART Host Read Command

The Compare command, Copy command, and Read command.

1.4.2.9 User Data Format

The layout of the data on the NVM media as described by the LBA Format of the namespace.

1.4.2.10 User Data Out Command

The Copy command and Write command.

1.4.3 Definitions specific to the NVM Command Set

This section defines terms that are specific to this specification.

1.4.3.1 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 145.

1.4.3.2 LBA range

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

1.4.3.3 logical block

The smallest addressable data unit for Read and Write commands.

1.4.3.4 logical block address (LBA)

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

1.5 Acronyms

Figure 2: Acronym definitions

Acronym	Definition
LBA	logical block address

1.6 References

NVM Express Base Specification, Revision 2.1. Available from https://www.nvmexpress.org.

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

NVM Express Subsystem Local Memory Command Set Specification, Revision 1.0. Available from https://www.nvmexpress.org.

NVM Express Zoned Namespace Command Set Specification, Revision 1.2. Available from https://www.nvmexpress.org.

SNIA® Solid State Storage (SSS) Performance Test Specification (PTS), Version 2.0.2, October 1, 2020. Available from https://www.snia.org.

1.6.1 References Under Development

None

1.6.2 Bibliography

INCITS 506-2021 SCSI Block Commands - 4 (SBC-4). Available from https://webstore.ansi.org.

INCITS 558-2021 ATA Command Set - 5 (ACS-5). Available from https://webstore.ansi.org.

TCG Storage Security Subsystem Class: Key Per IO Version 1.00 Revision 1.41. Available from https://trustedcomputinggroup.org.

2 NVM Command Set Model

The NVM Express Base Specification defines a property level interface for host software to communicate with a non-volatile memory subsystem (NVM subsystem). This specification defines additional functionality for the NVM Command Set.

2.1 Theory of operation

An NVM subsystem is comprised of some number of controllers, where each controller may access some number of namespaces. For the NVM Command Set, each namespace is comprised 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 data sizes may be 512 bytes, 1 KiB, 2 KiB, 4 KiB, 8 KiB, etc. NVM Command Set commands are used to access and modify logical block contents within a namespace.

2.1.1 Namespaces

A namespace is set of resources that may be accessed by a host and is as defined in the NVM Express Base Specification. A namespace has an associated namespace identifier that a host uses to access that namespace.

The Identify Namespace data structure (refer to Figure 114), for a namespace associated with this command set, contains related fields reporting the namespace size, namespace capacity, and namespace utilization:

- The Namespace Size (NSZE) field defines the total size of the namespace in logical blocks (LBA 0 through LBA n-1).
- The Namespace Capacity (NCAP) field defines the maximum number of logical blocks that may be allocated at any point in time.
- The Namespace Utilization (NUSE) field defines the number of logical blocks currently allocated in the namespace.

The following relationship holds: Namespace Size >= Namespace Capacity >= Namespace Utilization.

If the THINP bit is set to ‘1’ in the NSFEAT field of the Identify Namespace data structure, the controller:

- may report a value in the Namespace Capacity field that is less than the value in the Namespace Size field; and
- shall track the number of allocated blocks in the Namespace Utilization field.

If the THINP bit is cleared to ‘0’, the controller:

- shall report a value in the Namespace Capacity field that is equal to the value of Namespace Size field; and
- may report a value in the Namespace Utilization field that is always equal to the value in the Namespace Capacity field.

A logical block shall be marked as allocated when that logical block is written with:

- a User Data Out Command (refer to section 1.4.2.10);
- a Write Uncorrectable command (refer to section 3.3.7); or
- a Write Zeroes command (refer to section 3.3.8) that does not deallocate the logical block.

A logical block may be marked as allocated as the result of:

- a User Data Out Command not addressing the logical block (e.g., NPWG field may indicate sequential logical blocks placed and tracked together on the media (refer to section 5.2.2.1));
- a Write Zeroes command (refer to section 3.3.8) not addressing the logical block;

- a sanitize operation (refer to section 5.10); or.
- a Format NVM command (refer to section 4.1.2).

Commands and operations that may result in a logical block being deallocated include:

- a Dataset Management command (refer to section 3.3.3);
- a Write Zeroes command (refer to section 3.3.8) addressing the logical block;
- a sanitize operation (refer to section 5.10); or.
- a Format NVM command (refer to section 4.1.2).

Vendor specific means are able to allocate or deallocate logical blocks.

If the controller supports Asymmetric Namespace Access Reporting (i.e., the Asymmetric Namespace Access Reporting Support (ANARS) bit is set to '1' in the CMIC field in the Identify Controller data structure (refer to the NVM Express Base Specification)), then the NUSE field (refer to Figure 114) and the NVMCAP field (refer to Figure 114) are cleared to 0h if the relationship between the controller and the namespace is in the ANA Inaccessible state or the ANA Persistent Loss state (refer to the Asymmetric Namespace Access Reporting section in the NVM Express Base Specification). The Attached Namespace Attribute Changed asynchronous event and the Allocated Namespace Attribute Changed asynchronous event are not generated for changes to these fields that result from ANA state changes as described in the Asynchronous Event Request command section in the NVM Express Base Specification. The host uses the Asymmetric Namespace Access Change Notices as an indication of these changes.

2.1.2 Command Ordering Requirements

For all commands which are not part of a fused operation (refer to section 2.1.3), or for which the write size is greater than value indicated by the AWUN field (refer to section 2.1.4.2), 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 User Data In or User Data Out command to ensure any type of ordering between commands. For example, if a Read command is submitted for LBA *x* and there is a Write command also submitted for LBA *x*, there is no guarantee of the order of completion for those commands (the Read command may finish first or the Write command may finish first). If there are ordering requirements between these commands, host software or the associated application is required to enforce that ordering above the level of the controller.

2.1.3 Fused Operation

Fused operations are defined in the NVM Express Base Specification. The NVM Command Set adds the following requirements for Fused operations. The command sequences that may be used in a fused operation for the NVM Command Set are defined in Figure 3.

Figure 3: Supported Fused Operations

Command 1	Command 2	Fused Operation
Compare	Write	Compare and Write

2.1.3.1 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 operation 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 controller shall abort the Write command with a status code 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 command completion status is unaffected.

The LBA range, if used, shall be the same for the two commands. If the LBA ranges do not match, then the controller should abort the commands with a status code of Invalid Field in Command.

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 2.1.4) and that Atomic Boundaries are respected (refer to section 2.1.4.4). Controllers may abort a Compare and Write fused operation that is larger than ACWU/NACWU or that crosses an Atomic Boundary with a status code of Atomic Write Unit Exceeded.

2.1.4 Atomic Operation

A controller is in either Single Atomicity Mode (refer to section 2.1.4.1) or Multiple Atomicity Mode (refer to section 2.1.4.5) for each attached namespace. In Single Atomicity Mode, controller processing of a write command results in at most one atomic write operation, and controller processing of a write command that crosses any Namespace Atomic Boundary (refer to section 2.1.4.4) may or may not result in an atomic write operation. In Multiple Atomicity Mode, controller processing of a write command that crosses any Namespace Atomic Boundary (e.g., a write command that has a size greater than the AWUN/NAWUN value) results in multiple atomic write operations.

Multiple Atomicity Mode is a superset of Single Atomicity Mode. A host that is not aware of Multiple Atomicity Mode (i.e., is operating under the assumption that write commands are processed in Single Atomicity Mode) does not encounter unexpected or incompatible behavior.

2.1.4.1 Atomic Operation in Single Atomicity Mode

In Single Atomicity Mode, controller processing of a write command results in at most one atomic write operation. Figure 4 is an overview of the parameters that define the controller's support for Single Atomicity Mode atomic write operation. These parameters may affect command behavior and execution order based on write size (on a per controller or a per namespace basis).

Figure 4: Atomicity Parameters for Single Atomicity Mode

	Parameter Name	Value[1]
Controller Atomic Parameters (refer to Figure 117)	Atomic Write Unit Normal (AWUN)	
	Atomic Write Unit Power Fail (AWUPF)	≤ AWUN
	Atomic Compare and Write Unit (ACWU)	
Namespace Atomic Parameters (refer to the Identify Namespace data structure in Figure 114)	Namespace Atomic Write Unit Normal (NAWUN)	≥ AWUN
	Namespace Atomic Write Unit Power Fail (NAWUPF)	≥ AWUPF and ≤ NAWUN
	Namespace Atomic Compare and Write Unit (NACWU)	≥ ACWU
Namespace Atomic Boundary Parameters (refer to the Identify Namespace data structure in Figure 114)	Namespace Atomic Boundary Size Normal (NABSN)	≥ NAWUN and ≥ AWUN
	Namespace Atomic Boundary Offset (NABO)	≤ NABSN and ≤ NABSPF
	Namespace Atomic Boundary Size Power Fail (NABSPF)	≥ NAWUPF and ≥ AWUPF

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;
- the write portion of a Compare and Write fused operation (refer to section 2.1.3.1); and
- the write portion of a Copy command (refer to section 3.3.2).

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 atomicity parameters that are specific to the namespace and are indicated in the Identify Namespace data structure (refer to Figure 114). If an NVM subsystem reports a per namespace value, then that value shall be greater than or equal to the corresponding baseline value indicated in the Identify Controller data structure (refer to Figure 117).

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 the Identify Controller data structure or the Identify Namespace data structure 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 write operation. The Namespace Atomic Boundary Parameters (i.e., the NABSN, NABO, and NABSPF fields) define these boundaries for a namespace. A namespace supports Atomic Boundaries if NABSN field or NABSPF field is set to a non-zero value. For a namespace that does not support Atomic Boundaries, the controller 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 (e.g., a dispersed namespace or shared namespace that is not a dispersed 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.

2.1.4.2 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 that has a size less than or equal to the AWUN/NAWUN value and the write command does not cross an atomic boundary (refer to section 2.1.4.4), 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 that has a size greater than the AWUN/NAWUN value or crosses an atomic boundary, then:

- In Single Atomicity Mode (refer to section 2.1.4.1), there is no guarantee of command atomicity; and
- In Multiple Atomicity Mode (refer to section 2.1.4.5), atomicity is guaranteed for each portion of the command that falls within an atomic LBA subrange.

AWUN/NAWUN does not have any applicability to write errors caused by power failure or other error conditions (refer to section 2.1.4.3).

The host may indicate that AWUN and NAWUN are not necessary by configuring the Write Atomicity Normal feature (refer to section 4.1.3.3), which may result in higher performance in some implementations.

2.1.4.2.1 AWUN/NAWUN Example (Informative)

In this example, AWUN/NAWUN has a value of 2KiB (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 2KiB (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 5 shows valid results of the data in LBAs 0-4 and examples of invalid results (of which there are more possible combinations).

Figure 5: 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.

2.1.4.3 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 2.1.4.4), 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.

2.1.4.3.1 AWUPF/NAWUPF Example (Informative)

In this example, AWUPF/NAWUPF has a value of 1KiB (equivalent to two 512-byte logical blocks), AWUN/NAWUN has a value of 2KiB (equivalent to four 512-byte logical blocks) and the namespace atomic

boundary sizes (NABSN and NABSPF) are 0h. Command A writes LBAs 0 to 1. Figure 6 shows the initial state of the NVM.

Figure 6: 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 7 describes valid and invalid results.

Figure 7: 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.

2.1.4.3.2 Non-volatile requirements

After a write command has completed without error, reads for that location which are subsequently submitted and return data, shall return the data that was written by 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 the controller reports shutdown complete (CSTS.SHST set to 10b); and
e) main power loss occurs on a controller without completing the normal or abrupt shutdown procedure outlined in the Memory-based Transport Controller Shutdown or Message-based Transport Controller Shutdown sections in the NVM Express Base Specification,

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

2.1.4.4 Atomic Boundaries

Atomic Boundaries control how the atomicity guarantees defined in section 2.1.4 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 sizes shall be greater than or equal to the corresponding atomic write sizes:

- NABSN shall be greater than or equal to AWUN;
- NABSN shall be greater than or equal to NAWUN if NAWUN is reported;
- NABSPF shall be greater than or equal to AWUPF; and
- NABSPF shall be greater than or equal to NAWUPF if NAWUPF is reported.

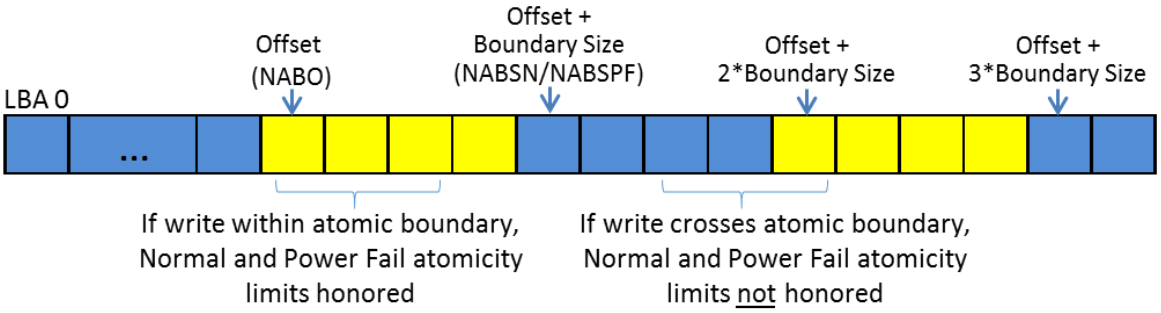
In addition, NABO shall be less than or equal to NABSN and NABSPF.

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 NABO and NABSN values, then for Single Atomicity Mode, the atomicity based on the NAWUN value is not guaranteed. If a write command crosses the atomic boundary specified by the NABO and NABSPF values, then for Single Atomicity Mode, the atomicity based on the NAWUPF value is not guaranteed. Atomicity guarantees for Multiple Atomicity Mode are specified in section 2.1.4.5.

Figure 8 shows an example of the behavior of Atomic Boundaries. Writes to an individual blue or yellow section do not cross an atomic boundary.

Figure 8: Atomic Boundaries Example



2.1.4.5 Atomic Operation in Multiple Atomicity Mode

In Multiple Atomicity Mode, controller processing of a write command that crosses any Namespace Atomic Boundary (e.g., a write command that has a size greater than the AWUN/NAWUN value) results in multiple atomic write operations.

Figure 9 is an overview of the Multiple Atomicity Mode controller parameters that differ from Single Atomicity Mode. These parameters may affect command behavior and execution order based on write size (on a per controller or a per namespace basis).

Figure 9: Atomicity Parameter Differences for Multiple Atomicity Mode

	Parameter Name	Value (Multiple Atomicity Mode)	Value (Single Atomicity Mode)
Namespace Atomic Boundary Parameters (refer to the Identify Namespace data structure in Figure 97)	Namespace Atomic Boundary Size Normal (NABSN)	= NAWUN, = AWUN if NAWUN is not reported, and = NABSPF	≥ NAWUN and ≥ AWUN
	Namespace Atomic Boundary Size Power Fail (NABSPF)	= NAWUPF, = AWUPF if NAWUPF is not reported, and = NABSN	≥ NAWUPF and ≥ AWUPF
	Namespace Atomic Boundary Offset (NABO)	≤ NABSN and ≤ NABSPF (same requirement for both modes)	

In Multiple Atomicity Mode, atomicity guarantees for a write command that does not cross a Namespace Atomic Boundary are the same as Single Atomicity Mode (refer to section 2.1.4.4).

In Multiple Atomicity Mode, controller processing of a write command that crosses any Namespace Atomic Boundary divides the command-specified range of LBAs into LBA subranges at Namespace Atomic Boundaries and performs an atomic write operation on each resulting LBA subrange, called an atomic LBA subrange. The atomicity guarantees apply separately to each atomic LBA subrange. The atomicity value requirements in Figure 9 ensure that each LBA in the command-specified LBA range is included in an atomic write operation (i.e., is part of an atomic LBA subrange).

For Multiple Atomicity Mode, atomicity parameter requirements depend on whether the NVM subsystem reports per namespace values for the atomicity parameters:

a. if per namespace values are reported, then the controller shall set the NABSN field, the NAWUN field, the NABSPF field, and the NAWUPF field to the same value; and
b. if per namespace values are not reported, then the controller shall set the NABSN field, the AWUN field, the NABSPF field, and the AWUPF field to the same value.

In Multiple Atomicity Mode, the write operation on each atomic LBA subrange is atomic to the NVM with respect to other read or write commands. This applies to both normal operating conditions and operation if a power fail or other error condition interrupts a write operation causing a torn write.

Multiple Atomicity Mode does not affect fused operations (refer to section 2.1.3). All write operations performed by a command that is part of a fused operation shall be performed in Single Atomicity Mode (refer to section 2.1.4.1).

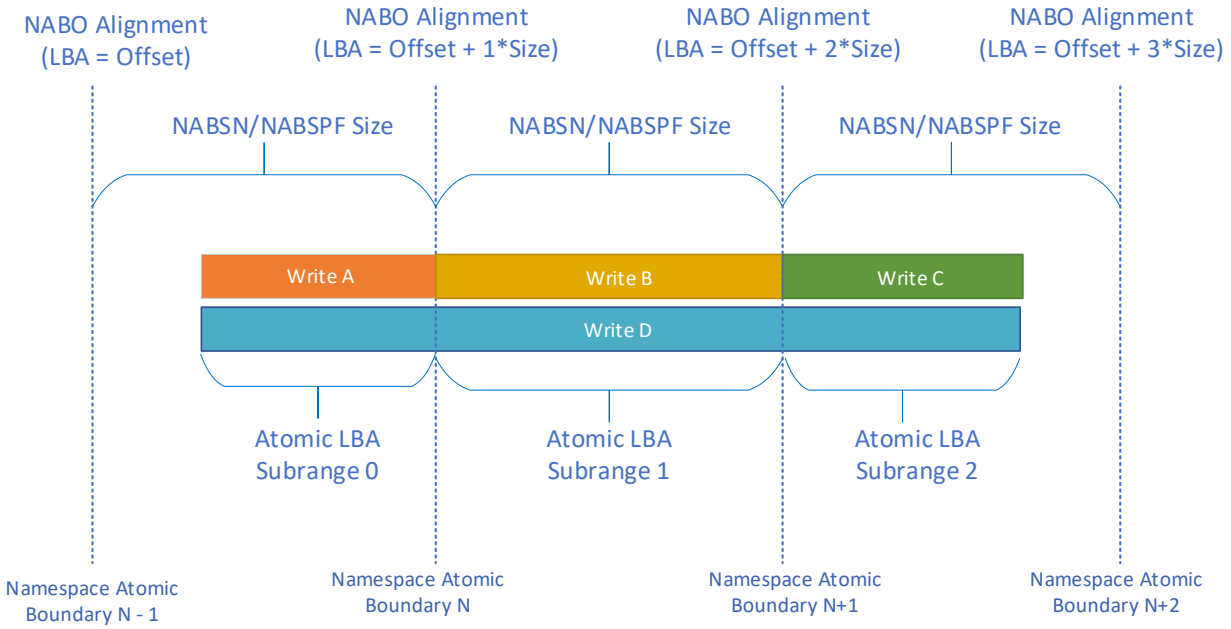
2.1.4.6 Namespace Atomic Boundaries in Single and Multiple Atomicity Modes

In Single Atomicity Mode write commands that cross Namespace Atomic Boundaries obtain no atomicity guarantees. For example, as shown in Figure 10, a single write command D that crosses Namespace Atomic Boundaries obtains no atomicity guarantees. Three separate write commands (A, B, and C) are necessary to obtain atomicity guarantees for the LBA subranges between Namespace Atomic Boundaries.

In contrast, in Multiple Atomicity Mode, write command D results in an atomicity guarantee for each LBA subrange obtained by dividing the LBA range at Namespace Atomic Boundaries (i.e., atomicity guarantees apply to LBA subranges 0, 1 and 2). In this Multiple Atomicity Mode example, a single write command (D)

in obtains atomicity guarantees that require three write commands (A, B and C) to obtain in Single Atomicity Mode.

Figure 10: Multiple Atomicity Example



2.1.5 End-to-end Protection Information

The NVM Command Set commands (refer to section 3.2) that include data transfer may utilize end-to-end data protection. Within these commands, the Protection Information Action, Protection Information Check, and Storage Tag Check fields are specified as defined in Figure 11 and Figure 12.

Figure 11: Protection Information Field Definition

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>03</td><td>Protection Information Action (PRACT): This bit specifies the action to take for the protection information. If the namespace is not formatted to use end-to-end protection information, then this bit shall be ignored by the controller. Refer to section 5.3.
<table>
<tr><th rowspan="2">PRACT Value</th><th colspan="2">Metadata Size</th><th rowspan="2">Definition</th></tr>
<tr><th>8B Protection Information Format</th><th>16B Protection Information Format</th></tr>
<tr><td>1b</td><td>8B</td><td>16B</td><td>The protection information is stripped (read) or inserted (write).</td></tr>
<tr><td>1b</td><td>> 8B</td><td>> 16B</td><td>The protection information is passed (read) or replaces the protection information in the metadata (write).</td></tr>
<tr><td>0b</td><td>Any</td><td>Any</td><td>The protection information is passed (read and write).</td></tr>
</table></td></tr>
</table>

Figure 11: Protection Information Field Definition

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>02:00</td><td>Protection Information Check (PRCHK): The protection information check field specifies the fields that shall be checked as part of end-to-end data protection processing. If the namespace is not formatted to use end-to-end protection information, then this field shall be ignored by the controller. Refer to section 5.3.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>02</td><td>Guard Check (GRDCHK): If this bit is set to ‘1’ enables protection information checking of the Guard field. If this bit is cleared to ‘0’, then the Guard field is not checked.</td></tr>
<tr><td>01</td><td>Application Tag Check (ATCHK): If this bit is set to ‘1’ enables protection information checking of the Application Tag field. If this bit is cleared to ‘0’, then the Application Tag field is not checked.</td></tr>
<tr><td>00</td><td>Reference Tag Check (RTCHK): If this bit is set to ‘1’ enables protection information checking of the Logical Block Reference Tag field. If this bit is cleared to ‘0’, then the Logical Block Reference Tag field is not checked.</td></tr>
</table></td></tr>
</table>

Figure 12: Storage Tag Check Definition

Bits	Description
00	**Storage Tag Check (STC):** This bit specifies the checking requirements for the Storage Tag field, if defined. If this bit is set to ‘1’, then protection information checking of the Storage Tag field is enabled. If this bit is cleared to ‘0’, then the Storage Tag field is not checked. Refer to section 5.3. If the Storage Tag Size (STS) field is cleared to 0h (refer to Figure 119), then this bit shall be ignored by the controller as no Storage Tag field is defined.

2.1.6 Metadata Region (MR)

Metadata may be supported for a namespace as part of the logical block (creating an extended logical block which is a larger logical block that is exposed to the application). Metadata may be transferred as interleaved with the logical block data (i.e., using the DPTR field) or as a separate buffer of data (i.e., using the MPTR field). The metadata shall not be split between the logical block data and a separate metadata buffer. For writes, the metadata shall be written atomically with its associated logical block. Refer to section 5.2.3.

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 and alignment of the Metadata Region is indicated by the Metadata Pointer field within the command.

The controller may support several physical formats of logical block data 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 (refer to section 5.3), then the last bytes of the metadata are used for protection information.

2.2 I/O Controller Requirements

2.2.1 Command Support

This specification implements the command support requirements for I/O controllers defined in the NVM Express Base Specification. Additionally, Figure 13 and Figure 14 define NVM Command Set specific definitions for commands that are mandatory, optional, and prohibited for an I/O controller that supports the NVM Command Set.

Figure 13: I/O Controller – Admin Command Support

Command	Combined Opcode Value	Command Support Requirements [1]	Reference
Get LBA Status	86h	O	4.2.1
Notes: O = Optional, M = Mandatory, P = Prohibited			

Figure 14: I/O Controller – NVM Command Set I/O Command Support

Command	Combined Opcode Value	Command Support Requirements [1]	Reference
Write	01h	M	3.3.6
Read	02h	M	3.3.4
Write Uncorrectable	04h	O	3.3.7
Compare	05h	O	3.3.1
Write Zeroes	08h	O	3.3.8
Verify	0Ch	O	3.3.5
Copy	19h	O	3.3.2
Vendor Specific	80h to FFh	O	
Notes: O = Optional, M = Mandatory, P = Prohibited			

2.2.2 Log Page Support

This specification implements the log page support requirements for I/O controllers defined in the NVM Express Base Specification. Additionally, Figure 15 defines NVM Command Set specific definitions for log pages that are mandatory, optional, and prohibited for an I/O controller that supports the NVM Command Set.

Figure 15: I/O Controller – NVM Log Page Support

Log Page Name	Log Page Identifier	Log Page Support Requirements [1]	Reference
LBA Status Information	0Eh	O	4.1.4.5
Notes: O = Optional, M = Mandatory, P = Prohibited, NR = Not Recommended			

2.2.3 Features Support

This specification implements the feature support requirements for I/O Controllers defined in the NVM Express Base Specification. Additionally, Figure 16 defines NVM Command Set specific definitions for features that are mandatory, optional, prohibited, and not recommended for an I/O Controller that supports the NVM Command Set.

Figure 16: I/O Controller – Feature Support

Feature Name	Feature Identifier	Feature Support Requirements [1]	Logged in Persistent Event Log	Reference
LBA Range Type	03h	O	NR	4.1.3.1
Error Recovery	05h	M	O	4.1.3.2
Write Atomicity Normal	0Ah	M	O	4.1.3.3
LBA Status Information Attributes	15h	O	O	4.1.3.5

Figure 16: I/O Controller – Feature Support

Feature Name	Feature Identifier	Feature Support Requirements [1]	Logged in Persistent Event Log	Reference
Performance Characteristics	1Ch	O	O	4.1.3.5
Notes: O = Optional, M = Mandatory, P = Prohibited, NR = Not Recommended				

3 I/O Commands for the NVM Command Set

This section specifies the NVM Command Set I/O commands.

3.1 Submission Queue Entry and Completion Queue Entry

The submission queue entry (SQE) structure and the fields that are common to all NVMe I/O Command Sets are defined in the Submission Queue Entry section in the NVM Express Base Specification. The completion queue entry (CQE) structure and the fields that are common to all NVMe I/O Command Sets are defined in the Completion Queue Entry section in the NVM Express Base Specification. The command specific fields in the SQE and CQE data structures (i.e., SQE Command Dword 2, Dword 3, Dwords 10-15 and CQE Dword 0, and Dword 1) for the NVM Command Set are defined in the following sections.

Completion queue entries indicate a Status Code Type (SCT) for the type of completion being reported. The status code type values and descriptions are described in the Status Field Definition section of the NVM Express Base Specification.

3.1.1 Common Command Format

The Common Command Format is as defined in the NVM Express Base Specification.

3.1.2 NVM Command Set Specific Status Values

Figure 17: Status Code – Generic Command Status Values

Value	Definition
14h	**Atomic Write Unit Exceeded:** The length specified exceeds the atomic write unit size.
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 least significant two bits of the Address or Length are not cleared to 00b. Note: An implementation compliant with revision 1.2.1 of the NVM Express Base Specification or earlier may use the status code value of 15h to indicate SGL Data Block Granularity Invalid.
25h	**Invalid Key Tag:** The command was denied due to an invalid KEYTAG (refer to section 5.4). Refer to the appropriate security specification (e.g., TCG Storage Interface Interactions specification).
80h	**LBA Out of Range:** The command references an LBA that exceeds the size of the namespace.
81h	**Capacity Exceeded:** The command requested an operation that exceeds the capacity of the namespace. This error occurs when the Namespace Utilization exceeds the Namespace Capacity, as reported in Figure 114.

Figure 18: Status Code – Command Specific Status Values

Value	Definition	Commands Affected
80h	Conflicting Attributes	Dataset Management, Read, Write
81h	Invalid Protection Information	Compare, Copy, Read, Verify, Write, Write Zeroes
82h	Attempted Write to Read Only Range	Copy, Dataset Management, Flush, Format NVM, Write, Write Uncorrectable, Write Zeroes
83h	Command Size Limit Exceeded	Copy, Dataset Management
85h	Incompatible Namespace or Format	Copy
86h	Fast Copy Not Possible	Copy
87h	Overlapping I/O Range	Copy
89h	Insufficient Resrouces	Copy
8Ah to BFh	Reserved	

Figure 19: Status Code – Media and Data Integrity Error Values

Value	Definition
85h	**Compare Failure:** The command failed due to a miscompare during a Compare command.
87h	**Deallocated or Unwritten Logical Block:** The command failed due to an attempt to copy from, read from, or verify an LBA range containing a deallocated or unwritten logical block.

3.2 I/O Command behavior for the NVM Command Set

This section defines specific behavior for I/O commands defined in the NVM Express Base Specification for the NVM Command Set.

3.2.1 I/O Management Receive command

3.2.1.1 Reclaim Unit Handle Status (Management Operation 01h)

The Reclaim Unit Handle Status Descriptor utilized for this management operation for the NVM Command Set is defined in Figure 20. The Reclaim Unit Handle Status Descriptor in the Reclaim Unit Handle Status Descriptor List shall be listed first in ascending order of Placement Handle and second in ascending order of Reclaim Group Identifier.

Figure 20: Reclaim Unit Handle Status Descriptor

Bytes	Description
1:0	**Placement Identifier (PID):** This field indicates the Placement Identifier (refer to the Reclaim Unit Handle Status section in the NVM Express Base Specification) containing the Placement Handle and Reclaim Group Identifier for this Reclaim Unit Handle Status Descriptor.
3:2	**Reclaim Unit Handle Identifier (RUHID)**: This field indicates the Reclaim Unit Handle for the Placement Identifier field.
7:4	**Estimated Active Reclaim Unit Time Remaining (EARUTR):** This field indicates an estimate of the time in seconds that the Reclaim Unit currently referenced by the Reclaim Unit Handle is allowed to remain referenced by that Reclaim Unit Handle (refer to the Flexible Data Placement section in the NVM Express Base Specification) before the controller may modify the Reclaim Unit Handle to reference a different Reclaim Unit. This value is the remaining time at the time the I/O Management Receive command is processed by the controller. If this field is cleared to 0h, then no time is reported.
15:08	**Reclaim Unit Available Media Writes (RUAMW):** This field indicates the number of logical blocks which are currently able to be written to the media associated with the Reclaim Unit currently referenced by the Placement Identifier field. The product of this field and the Formatted LBA Size field is able to be less than the nominal size (refer to the RUNS field in the FDP Configurations log page defined in the NVM Express Base Specification) of the Reclaim Unit (e.g., the Reclaim Unit has not been written, but excess defects prevent writing some of the media). The product of this field and the Formatted LBA Size field is able to be greater than the nominal size (refer to the RUNS field in the FDP Configurations log page defined in the NVM Express Base Specification) of the Reclaim Unit (e.g., there is over-provisioned capacity to support the Reclaim Unit). The value of this field may or may not be modified by a Controller Level Reset or the processing of a Flush command.
31:16	Reserved

3.3 NVM Command Set Commands

The NVM Command Set includes the commands listed in Figure 21. This section describes the definition for each of the commands defined by this specification. Commands are submitted as described in the NVM Express Base Specification. Physical region page (PRP) entries (refer to the Data Layout section of the

NVM Express Base Specification) and scatter gather lists (SGL) (refer to the Data Layout section of the NVM Express Base Specification) are used by the NVM Command Set commands to describe data buffers.

In the case of Compare, Read, Verify, 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) bit in the command. The time limit is specified in the Error Recovery feature, specified in section 4.1.3.5. 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.

If a host does not set the LBA Format Extension Enable (LBAFEE) field to 1h in the Host Behavior Support feature (refer to section 4.1.3.6), then a controller aborts all I/O commands that access user data to namespaces formatted with (refer to section 5.3.1):

a) 16b Guard Protection Information with the STS field set to a non-zero value;
b) 32b Guard Protection Information; and
c) 64b Guard Protection Information.

Figure 21: Opcodes for NVM Commands

Opcode by Field		Combined Opcode [1]	Command [2]	Reference Section
(07:02)	(01:00)			
Function	Data Transfer [3]			
0000 00b	00b	00h	Flush [4]	NVM Express Base Specification
0000 00b	01b	01h	Write	3.3.6
0000 00b	10b	02h	Read	3.3.4
0000 01b	00b	04h	Write Uncorrectable	3.3.7
0000 01b	01b	05h	Compare	3.3.1
0000 10b	00b	08h	Write Zeroes	3.3.8
0000 10b	01b	09h	Dataset Management	3.3.3
0000 11b	00b	0Ch	Verify	3.3.5
0000 11b	01b	0Dh	Reservation Register	NVM Express Base Specification
0000 11b	10b	0Eh	Reservation Report	
0001 00b	01b	11h	Reservation Acquire	
0001 00b	10b	12h	I/O Management Receive	
0001 01b	01b	15h	Reservation Release	
0001 10b	00b	18h	Cancel [4]	
0001 10b	01b	19h	Copy	3.3.2
0001 11b	01b	1Dh	I/O Management Send	NVM Express Base Specification
Vendor Specific				
n/a	NOTE 3	80h to FFh	Vendor specific	

Key:
Base = NVM Express Base Specification

Notes:
1. Opcodes not listed are reserved.
2. All NVM commands use the Namespace Identifier (NSID) field. The value FFFFFFFFh is not supported in this field unless footnote 4 in this figure indicates that a specific command does support that value.
3. Indicates the data transfer direction of the command. All options to the command shall transfer data as specified or transfer no data. All commands, including vendor specific commands, shall follow this convention: 00b = no data transfer; 01b = host to controller; 10b = controller to host; 11b = bidirectional.
4. This command may support the use of the Namespace Identifier (NSID) field set to FFFFFFFFh.

3.3.1 Compare command

The Compare command reads the logical blocks specified by the command from the non-volatile 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. The command may specify protection information to be checked as described in section 5.3.2.4.

The command uses Command Dword 2, Command Dword 3, 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. All other command specific fields are reserved.

Figure 22: Compare – Metadata Pointer

Bits	Description
63:00	**Metadata Pointer (MPTR):** This field contains the Metadata Pointer, if applicable. Refer to the Common Command Format figure in the NVM Express Base Specification for the definition of this field.

Figure 23: Compare – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the data to use for the compare. Refer to the Common Command Format figure in the NVM Express Base Specification for the definition of this field.

Figure 24: Compare – Command Dword 2 and Dword 3

Bits	Description
63:48	Reserved
47:00	**Expected Logical Block Tags Upper (ELBTU):** This field and Command Dword 14 specify the variable sized Expected Logical Block Storage Tag (ELBST) and Expected Initial Logical Block Reference Tag (EILBRT) fields, which are defined in section 5.3.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller.

Figure 25: Compare – Command Dword 10 and Command Dword 11

Bits	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 26: Compare – Command Dword 12

Bits	Description
31	**Limited Retry (LR):** If this bit is set to '1', then the controller should apply limited retry efforts. If this bit is cleared to '0', then the controller should apply all available error recovery means to retrieve the data for comparison.

Figure 26: Compare – Command Dword 12

Bits	Description
30	**Force Unit Access (FUA):** If this bit is set to '1', then for data and metadata, if any, associated with logical blocks specified by the Compare command, the controller shall: 1) commit that data and metadata, if any, to non-volatile medium; and 2) read the data and metadata, if any, from non-volatile medium. If this bit is cleared to '0', then this bit has no effect.
29:26	**Protection Information (PRINFO):** Specifies the protection information action and check field, as defined in Figure 11. The Protection Information Action (PRACT) bit shall be cleared to '0'. If the Protection Information Check (PRCHK) field is non-zero, protection checks are performed on the logical blocks transferred from the host and on the logical blocks read from NVM (refer to section 5.3.2.4).
25	Reserved
24	**Storage Tag Check (STC):** This bit specifies the Storage Tag field shall be checked as part of end-to-end data protection processing as defined in Figure 12.
23:20	Reserved
19:16	**Command Extension Type (CETYPE):** Specifies the Command Extension Type that applies to the command (refer to the Key Per I/O section in the NVM Express Base Specification).
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.

The definition of Command Dword 13 is based on the CETYPE value. If the CETYPE value is cleared to 0h, then Command Dword 13 is reserved. If the CETYPE value is non-zero, then Command Dword 13 is defined in Figure 27.

Figure 27: Compare - Command Dword 13 if CETYPE is non-zero

Bits	Description
31:16	Reserved
15:00	**Command Extension Value (CEV):** The definition of this field is dependent on the value of the CETYPE field. Refer to the Key Per I/O section in the NVM Express Base Specification.

Figure 28: Compare – Command Dword 14

Bits	Description
31:00	**Expected Logical Block Tags Lower (ELBTL):** This field and bits 47:00 of Command Dword 2 and Dword 3 specify the variable sized Expected Logical Block Storage Tag (ELBST) and Expected Initial Logical Block Reference Tag (EILBRT) fields, which are defined in section 5.3.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller.

Figure 29: Compare – Command Dword 15

Bits	Description
31:16	**Expected Logical Block Application Tag Mask (ELBATM):** This field specifies the Application Tag Mask expected value. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.
15:00	**Expected Logical Block Application Tag (ELBAT):** This field specifies the Application Tag expected value. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.

3.3.1.1 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 30.

Figure 30: Compare – Command Specific Status Values

Value	Definition
81h	**Invalid Protection Information:** The Protection Information (PRINFO) field (refer to Figure 26) settings specified in the command are invalid for the Protection Information with which the namespace was formatted (refer to the PI field in Figure 90 and the DPS field in Figure 114) or the EILBRT field is invalid (refer to section 5.3.3).

3.3.2 Copy command

The Copy command is used by the host to copy user data from one or more source ranges in one or more source namespaces to a single consecutive destination logical block range in a destination namespace (i.e., the namespace specified by the NSID field). Each source range may be in the same namespace or a different namespace with respect to any other source range and with respect to the destination logical block range.

The command uses Command Dword 2, Command Dword 3, 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 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 31: Copy – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the data to use for the command. Refer to the Common Command Format figure in the NVM Express Base Specification for the definition of this field.

Figure 32: Copy – Command Dword 2 and Dword 3

Bits	Description
63:48	Reserved
47:00	**Logical Block Tags Upper (LBTU):** This field and Command Dword 14 specify the variable sized Logical Block Storage Tag (LBST) and Initial Logical Block Reference Tag (ILBRT) fields, which are defined in section 5.3.1.4.1, to be used for the write portion of the copy operation. If the destination namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller.

Figure 33: Copy – Command Dword 10 and Command Dword 11

Bits	Description
63:00	**Starting Destination LBA (SDLBA):** This field indicates the 64-bit address of the first logical block to be written as part of the copy operation. Command Dword 10 contains bits 31:00; Command Dword 11 contains bits 63:32.

Figure 34: Copy – Command Dword 12

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31</td><td>Limited Retry (LR): If this bit is set to ‘1’, then the controller should apply limited retry efforts for the write portion of the copy operation. If this bit is cleared to ‘0’, then the controller should apply all available error recovery means to write the data to the NVM.</td></tr>
<tr><td>30</td><td>Force Unit Access (FUA): If this bit is set to ‘1’, then for data and metadata, if any, associated with logical blocks specified by the write portion of the copy operation, the controller shall write that data and metadata, if any, to non-volatile medium before indicating command completion.

There is no implied ordering with other commands. If this bit is cleared to ‘0’, then this bit has no effect.</td></tr>
<tr><td>29:26</td><td>Protection Information Write (PRINFOW): Specifies the protection information action and check field, as defined in Figure 11, to be used for the write portion of the copy operation.</td></tr>
<tr><td>25</td><td>Storage Tag Check Read (STCR): This bit specifies the Storage Tag field shall be checked as part of end-to-end data protection processing as defined in Figure 12, to be used for the read portion of the copy operation. If the Storage Tag Check Read Support (STCRS) bit (refer to Figure 111) is cleared to ‘0’, then this bit is reserved.</td></tr>
<tr><td>24</td><td>Storage Tag Check Write (STCW): This bit specifies the Storage Tag field shall be checked as part of end-to-end data protection processing as defined in Figure 12, to be used for the write portion of the copy operation.</td></tr>
<tr><td>23:20</td><td>Directive Type (DTYPE): Specifies the Directive Type associated with the Directive Specific field (refer to the Directives section in the NVM Express Base Specification) used for the write portion of the copy operation.</td></tr>
<tr><td>19:16</td><td>Command Extension Type (CETYPE): Specifies the Command Extension Type that applies to the command (refer to the Key Per I/O section in the NVM Express Base Specification). This field is used for the write portion of the copy operation.</td></tr>
<tr><td>15:12</td><td>Protection Information Read (PRINFOR): Specifies the protection information action and check field, as defined in Figure 11, to be used for the read portion of the copy operation specified by each Source Range entries.</td></tr>
<tr><td>11:08</td><td>Descriptor Format (DESFMT): Specifies the type of the Copy Descriptor Format that is used. The Copy Descriptor Format specifies the starting location, length, and parameters associated with the read portion of the operation.
<table>
<tr><th>Code Descriptor Format Type</th><th>Definition</th></tr>
<tr><td>0h</td><td>Source Range Entries Copy Descriptor Format 0h is used (refer to Figure 39).</td></tr>
<tr><td>1h</td><td>Source Range Entries Copy Descriptor Format 1h is used (refer to Figure 40).</td></tr>
<tr><td>2h</td><td>Source Range Entries Copy Descriptor Format 2h is used (refer to Figure 39).</td></tr>
<tr><td>3h</td><td>Source Range Entries Copy Descriptor Format 3h is used (refer to Figure 40).</td></tr>
<tr><td>4h</td><td>Source Range Entries Copy Descriptor Format 4h is used (refer to NVM Express Subsystem Local Memory Command Set Specification).</td></tr>
<tr><td>All Others</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>07:00</td><td>Number of Ranges (NR): Specifies the number of Source Range entries that are specified in the command. This is a 0’s-based value.</td></tr>
</table>

Figure 35: Copy – Command Dword 13

Bits	Description
31:16	**Directive Specific (DSPEC)**: Specifies the Directive Specific value associated with the Directive Type field (refer to the Directives section in the NVM Express Base Specification).
15:00	**Command Extension Value (CEV):** The definition of this field is dependent on the value of the CETYPE field. Refer to the Key Per I/O section in the NVM Express Base Specification. This field is used for the write portion of the copy operation.

Figure 36: Copy – Command Dword 14

Bits	Description
31:00	**Logical Block Tags Lower (LBTL):** This field and bits 47:00 of Command Dword 2 and Dword 3 specify the variable sized Logical Block Storage Tag (LBST) and Initial Logical Block Reference Tag (ILBRT) fields, which are defined in section 5.3.1.4.1, to be used for the write portion of the copy operation. If the destination namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller.

Figure 37: Copy – Command Dword 15

Bits	Description
31:16	**Logical Block Application Tag Mask (LBATM):** This field specifies the Application Tag Mask value for the write portion of the copy operation. If the destination namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.
15:00	**Logical Block Application Tag (LBAT):** This field specifies the Application Tag value for the write portion of the copy operation. If the destination namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.

The controller shall indicate the Source Range Entries Copy Descriptor Formats supported by the controller in the Copy Descriptor Formats Supported field in the Identify Controller data structure (refer to the NVM Express Base Specification).

Controller usage of Source Range Entries Copy Descriptor Formats 2h and 3h is further qualified by whether the host has enabled these formats in the Host Behavior Support feature (refer to the NVM Express Base Specification). If the controller supports a Source Range Entries Copy Descriptor Format that has not been enabled, the controller shall process Copy commands as if that format is not supported (e.g., if that format is specified in the Descriptor Format field in Command Dword 12, the controller shall abort the command with a status code of Invalid Field in Command). Source Range Entries Copy Descriptor Formats 0h and 1h are always enabled if supported. A host that enables Source Range Entries Copy Descriptor Formats 2h and/or 3h indicates to the controller that the host accepts the implications (e.g., for namespace access control) of the presence of a Source Namespace Identifier (SNSID) in these formats (refer to Figure 39 and Figure 40).

The data that the Copy command provides is a list of Source Range entries that describe the data to be copied to the destination range starting at the LBA specified by the SDLBA field. The Copy Descriptor Format type of the Source Range entries is specified in the Descriptor Format field in Command Dword 12. The Copy Descriptor Format types are distinguished by the supported protection information formats (refer to section 5.3.1) and whether the Copy Descriptor Format contains a Source Namespace Identifier (SNSID) field that supports a copy source in a different namespace than the copy destination, as described in Figure 38. For a Copy Descriptor Format that does not contain an SNSID field, the source namespace is the same as the destination namespace which is specified by the NSID field in the command.

Figure 38: Copy – Copy Descriptor Formats

Copy Descriptor Format type	Protection Information Formats	SNSID field present	Description
0h	16b Guard Protection Information	No	Protection Information size: 8 bytes, source namespace and destination namespace are the same.

Figure 38: Copy – Copy Descriptor Formats

Copy Descriptor Format type	Protection Information Formats	SNSID field present	Description
1h	32b Guard Protection Information 64b Guard Protection Information	No	Protection Information size: 16 bytes, source namespace and destination namespace are the same.
2h	16b Guard Protection Information	Yes	Protection Information size: 8 bytes, source namespace may differ from destination namespace.
3h	32b Guard Protection Information 64b Guard Protection Information	Yes	Protection Information size: 16 bytes, source namespace may differ from destination namespace.
4h	None	Yes	The source namespace supports the Subsystem Local Memory (SLM) Command Set (refer to the NVM Express Subsystem Local Memory Command Set Specification).

If the Copy Descriptor Format specified in the Descriptor Format field is not supported by the controller, then the command shall be aborted with a status code of Invalid Field in Command.

If:

a) the Copy Descriptor Format specified in the Descriptor Format field is supported by the controller;
b) the specified destination namespace is formatted to use 16b Guard Protection Information; and
c) the Descriptor Format field is not cleared to 0h and is not set to 2h,

then the command shall be aborted with a status code of Invalid Namespace or Format.

If:

a) the Copy Descriptor Format specified in the Descriptor Format field is supported by the controller;
b) the specified destination namespace is formatted to use 32b Guard Protection Information or 64b Guard Protection Information; and
c) the Descriptor Format field is not set to 1h and is not set to 3h,

then the command shall be aborted with a status code of Invalid Namespace or Format.

Figure 39 shows the Copy Descriptor Format 0h and Format 2h descriptors with an example that has 128 Source Range entries.

Figure 39: Copy – Source Range Entries Copy Descriptor Format 0h and Format 2h

<table>
<tr><th>Range</th><th>Bytes</th><th>Description</th></tr>
<tr><td rowspan="3">Source Range 0</td><td>03:00</td><td>Source Parameters (SPARS): This field specifies attributes as follows:
<table>
<tr><th>Format 0h bytes</th><th>Format 2h bytes</th><th>Description</th></tr>
<tr><td>03:00</td><td>n/a</td><td>Reserved</td></tr>
<tr><td>n/a</td><td>03:00</td><td>Source Namespace Identifier (SNSID): Specifies the source namespace for this Source Range entry.</td></tr>
</table></td></tr>
<tr><td>07:04</td><td>Reserved</td></tr>
<tr><td>15:08</td><td>Starting LBA (SLBA)</td></tr>
</table>

Figure 39: Copy – Source Range Entries Copy Descriptor Format 0h and Format 2h

<table>
<tr><th>Range</th><th>Bytes</th><th>Description</th></tr>
<tr><td rowspan="2"></td><td>19:16</td><td>Read Parameters (RPARS): This field specifies attributes as follows.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:20</td><td>Reserved</td></tr>
<tr><td>19:16</td><td>Command Extension Type (CETYPE): Specifies the Command Extension Type that applies to the command (refer to the Key Per I/O section in the NVM Express Base Specification). This field is used for the read portion of the copy operation for the LBAs specified in this Source Range entry.</td></tr>
<tr><td>15:00</td><td>Number of Logical Blocks (NLB): This field indicates the number of logical blocks to be copied. This is a 0’s based value.</td></tr>
</table></td></tr>
<tr><td>21:20</td><td>Command Extension Value (CEV): The definition of this field is dependent on the value of the CETYPE field. Refer to the Key Per I/O section in the NVM Express Base Specification. This field is used for the read portion of the copy operation for the LBAs specified in this Source Range entry.</td></tr>
<tr><td rowspan="2"></td><td rowspan="2">23:22</td><td rowspan="2">Source Options (SOPT): This field specifies options as follows:
<table>
<tr><th>Format 0h bits</th><th>Format 2h bits</th><th>Description</th></tr>
<tr><td>n/a</td><td>15</td><td>Fast Copy Only (FCO): If this bit is set to ‘1’, then the controller only performs fast copy operations (refer to section 3.3.2.1) for user data in this Source Range entry. If this bit is cleared to ‘0’, then this bit has no effect.</td></tr>
<tr><td>15:00</td><td>14:00</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td rowspan="3"></td></tr>
<tr><td>27:24</td><td>Expected Logical Block Tags (ELBT): This field specifies the variable sized Expected Logical Block Storage Tag (ELBST) and Expected Initial Logical Block Reference Tag (EILBRT), which are defined in section 5.3.1.4.1, to be used for the read portion of the copy operation for the LBAs specified in this Source Range entry. If the source namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.</td></tr>
<tr><td>29:28</td><td>Expected Logical Block Application Tag (ELBAT): This field specifies the Application Tag expected value used for the read portion of the copy operation for the LBAs specified in this Source Range entry. If the source namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.</td></tr>
<tr><td>31:30</td><td>Expected Logical Block Application Tag Mask (ELBATM): This field specifies the Application Tag Mask expected value used for the read portion of the copy operation for the LBAs specified in this Source Range entry. If the source namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td rowspan="10">Source Range 1</td><td>35:32</td><td>Source Parameters (SPARS)</td></tr>
<tr><td>39:36</td><td>Reserved</td></tr>
<tr><td>47:40</td><td>Starting LBA (SLBA)</td></tr>
<tr><td>51:48</td><td>Read Parameters (RPARS)</td></tr>
<tr><td>53:52</td><td>Command Extension Type (CEV)</td></tr>
<tr><td>55:54</td><td>Source Options (SOPT)</td></tr>
<tr><td>59:56</td><td>Expected Logical Block Tags (ELBT): (i.e., ELBST and EILBRT)</td></tr>
<tr><td>61:60</td><td>Expected Logical Block Application Tag (ELBAT)</td></tr>
<tr><td>63:62</td><td>Expected Logical Block Application Tag Mask (ELBATM)</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td rowspan="8">Source Range 127</td><td>4067:4064</td><td>Source Parameters (SPARS)</td></tr>
<tr><td>4071:4068</td><td>Reserved</td></tr>
<tr><td>4079:4072</td><td>Starting LBA (SLBA)</td></tr>
<tr><td>4083:4080</td><td>Read Parameters (RPARS)</td></tr>
<tr><td>4085:4084</td><td>Command Extension Type (CEV)</td></tr>
<tr><td>4087:4086</td><td>Source Options (SOPT)</td></tr>
</table>

Figure 39: Copy – Source Range Entries Copy Descriptor Format 0h and Format 2h

Range	Bytes	Description
	4091:4088	**Expected Logical Block Tags (ELBT):** (i.e., ELBST and EILBRT)
	4093:4092	**Expected Logical Block Application Tag (ELBAT)**
	4095:4094	**Expected Logical Block Application Tag Mask (ELBATM)**

The SNSID field (refer to Figure 39) specifies an active NSID that identifies the namespace for the source range. If the SNSID field contains an invalid NSID, the value 0h or the value FFFFFFFFh, then the controller shall abort the Copy command with a status code of Invalid Namespace or Format. If the SNSID field contains an inactive NSID, then the controller shall abort the Copy command with a status code of Invalid Field in Command.

Figure 40 shows the Copy Descriptor Format 1h and Format 3h descriptors with an example that has 102 Source Range entries.

Figure 40: Copy – Source Range Entries Copy Descriptor Format 1h and Format 3h

<table>
<tr><th>Range</th><th>Bytes</th><th>Description</th></tr>
<tr><td rowspan="6">Source Range 0</td><td>03:00</td><td>Source Parameters (SPARS): This field specifies attributes as follows:
<table>
<tr><th>Format 1h bytes</th><th>Format 3h bytes</th><th>Description</th></tr>
<tr><td>03:00</td><td>n/a</td><td>Reserved</td></tr>
<tr><td>n/a</td><td>03:00</td><td>Source Namespace Identifier (SNSID): Specifies the source namespace for this Source Range entry.</td></tr>
</table></td></tr>
<tr><td>07:04</td><td>Reserved</td></tr>
<tr><td>15:08</td><td>Starting LBA (SLBA)</td></tr>
<tr><td>19:16</td><td>Read Parameters (RPARS): This field specifies attributes as follows:
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:20</td><td>Reserved</td></tr>
<tr><td>19:16</td><td>Command Extension Type (CETYPE): Specifies the Command Extension Type that applies to the command (refer to the Key Per I/O section in the NVM Express Base Specification). This field is used for the read portion of the copy operation for the LBAs specified in this Source Range entry.</td></tr>
<tr><td>15:00</td><td>Number of Logical Blocks (NLB): This field indicates the number of logical blocks to be copied. This is a 0’s based value.</td></tr>
</table></td></tr>
<tr><td>21:20</td><td>Command Extension Value (CEV): The definition of this field is dependent on the value of the CETYPE field. Refer to the Key Per I/O section in the NVM Express Base Specification. This field is used for the read portion of the copy operation for the LBAs specified in this Source Range entry.</td></tr>
<tr><td>23:22</td><td>Source Options (SOPT): This field specifies options as follows:
<table>
<tr><th>Format 1h bits</th><th>Format 3h bits</th><th>Description</th></tr>
<tr><td>n/a</td><td>15</td><td>Fast Copy Only (FCO): If this bit is set to ‘1’, then the controller only performs fast copy operations (refer to section 3.3.2.1) for user data in this Source Range entry. If this bit is cleared to ‘0’, then this bit has no effect.</td></tr>
<tr><td>15:00</td><td>14:00</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>25:24</td><td>Reserved</td></tr>
</table>

Figure 40: Copy – Source Range Entries Copy Descriptor Format 1h and Format 3h

Range	Bytes	Description
	35:26	**Expected Logical Block Tags (ELBT):** This field specifies variable sized Expected Logical Block Storage Tag (ELBST) and Expected Initial Logical Block Reference Tag (EILBRT) fields, which are defined in section 5.3.1.4.1, to be used for the read portion of the copy operation. If the source namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.
	37:36	**Expected Logical Block Application Tag (ELBAT):** This field specifies the Application Tag expected value used for the read portion of the copy operation for the LBAs specified in this Source Range entry. If the source namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.
	39:38	**Expected Logical Block Application Tag Mask (ELBATM):** This field specifies the Application Tag Mask expected value used for the read portion of the copy operation for the LBAs specified in this Source Range entry. If the source namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.
Source Range 1	43:40	**Source Parameters (SPARS)**
	47:44	Reserved
	55:48	**Starting LBA (SLBA)**
	59:56	**Read Parameters (RPARS)**
	61:60	**Command Extension Value (CEV)**
	63:62	**Source Options (SOPT)**
	65:64	Reserved
	75:66	**Expected Logical Block Tags (ELBT):** (i.e., ELBST and EILBRT)
	77:76	**Expected Logical Block Application Tag (ELBAT)**
	79:78	**Expected Logical Block Application Tag Mask (ELBATM)**
		…
Source Range 101	4043:4040	**Source Parameters (SPARS)**
	4047:4044	Reserved
	4055:4048	**Starting LBA (SLBA)**
	4059:4056	**Read Parameters (RPARS)**
	4061:4060	**Command Extension Value (CEV)**
	4063:4062	**Source Options (SOPT)**
	4065:4064	Reserved
	4075:4066	**Expected Logical Block Tags (ELBT)** (i.e., ELBST and EILBRT)
	4077:4076	**Expected Logical Block Application Tag (ELBAT)**
	4079:4078	**Expected Logical Block Application Tag Mask (ELBATM)**

The SNSID field (refer to Figure 40) specifies an active NSID that identifies the namespace for the source range. If the SNSID field contains an invalid NSID, the value 0h, or the value FFFFFFFFh, then the controller shall abort the Copy command with a status code of Invalid Namespace or Format. If the SNSID field contains an inactive NSID, then the controller shall abort the Copy command with a status code of Invalid Field in Command.

If the number of Source Range entries (i.e., the value in the NR field) is greater than the value in the MSRC field (refer to Figure 114), then the Copy command shall be aborted with a status code of Command Size Limit Exceeded.

For an LBA based Copy Descriptor Format Type (i.e., 0h, 1h, 2h, or 3h):

- the number of logical blocks written by the Copy command is the sum of all Number of Logical Blocks fields in all Source Range entries specified in the list of Source Range entries;

- if a valid Source Range entry specifies a Number of Logical Blocks field that is greater than the value in the MSSRL field (refer to Figure 114), then the Copy command shall be aborted with a status code of Command Size Limit Exceeded; and
- if the sum of all Number of Logical Blocks fields in all Source Range entries is greater than the value in the MCL field (refer to Figure 114), then the Copy command shall be aborted with a status code of Command Size Limit Exceeded.

For a byte based Copy Descriptor Format Type (i.e., 4h):

- the number of logical blocks written by the Copy command is determined by the sum of the Number of Bytes fields (refer to the NVM Express Subsystem Local Memory Command Set Specification) in all Source Range entries specified in the list of Source Range entries divided by the LBA Data Size field (refer to Figure 115) of the LBA Format data structure associated with the destination namespace);
- if the sum of all Number of Bytes fields in all Source Range entries does not represent a multiple of the LBA Data Size field of the destination namespace, then the Copy command shall be aborted with a status code of Invalid Field in Command;
- if a valid Source Range entry specifies a Number of Bytes field that represents a number of logical blocks (i.e., as determined by the LBA Data Size field) that is greater than the value in the MSSRL field (refer to Figure 114), then the Copy command shall be aborted with a status code of Command Size Limit Exceeded; and
- if the sum of all Number of Bytes fields in all Source Range entries represents a number of logical blocks (i.e., as determined by the LBA Data Size field) that is greater than the value in the MCL field (refer to Figure 114), then the Copy command shall be aborted with a status code of Command Size Limit Exceeded.

The data bytes in the LBAs specified by each Source Range entry shall be copied to the destination LBA range in the same order those LBAs are listed in the Source Range entries (e.g., the LBAs specified by Source Range 0 are copied to the lowest numbered LBAs specified by the SDLBA field, the LBAs specified by Source Range 1 are copied to the next consecutively numbered LBAs after the LBAs copied for Source Range entry 0). The read operations and write operations used to perform the copy may operate sequentially or in parallel.

The host should not specify a destination LBA range that overlaps the LBA in any of the Source Range entries. If the host specifies a destination LBA range that overlaps with any LBAs specified in one or more of the Source Range entries, then upon completion of the Copy command, the data stored in each logical block in that overlapping destination LBA range may, within the constraints of the atomicity rules described in section 2.1.4, be from any of the one or more Source Range entries in which that LBA is contained. This is a result of the possibility that overlapping Source Range entries may be processed in any order

Two LBA ranges overlap if they specify LBAs in the same namespace and there is at least one LBA that is part of both LBA ranges.

For Source Range Entries Copy Descriptor Formats 0h and 1h, the host should not specify a destination LBA range that overlaps the LBA in any of the Source Range entries. If the host specifies a destination LBA range that overlaps with any LBAs specified in one or more of the Source Range entries, then upon completion of the Copy command, the data stored in each logical block in that overlapping destination LBA range may, within the constraints of the atomicity rules described in section 2.1.4, be from any of the one or more Source Range entries in which that LBA is contained. This is a result of the possibility that overlapping Source Range entries may be processed in any order.

For Source Range Entries Copy Descriptor Formats 2h and 3h, overlap of any source LBA range that is located in the destination namespace with the destination LBA range is prohibited. If a Copy command uses either Source Range Entries Copy Descriptor Format 2h or 3h and any specified source LBA range that is located in the destination namespace has any LBAs in common with the specified destination LBA range, then the controller shall abort the command with a status code of Overlapping I/O Range.

If the controller:

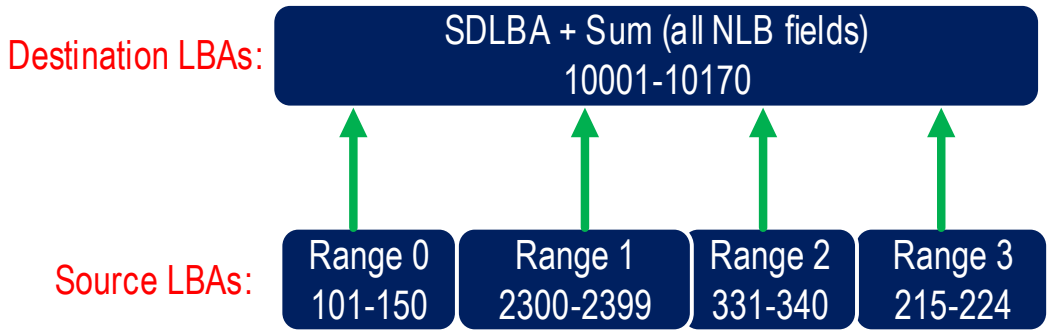
- supports reachability reporting (refer to the Reachability Reporting architecture section in the NVM Express Base Specification);
- processes a Copy command that requests a copy between different namespaces (i.e., the NSID in the SNSID field in a Source Range entry is different than the NSID of the destination namespace); and
- does not report a Reachability Association (refer to the Reachability Reporting architecture section in the NVMe Base Specification) between those namespaces,

then the controller shall abort the command with the status code set to Namespace Not Reachable.

If the read portion of a copy operation attempts to access a deallocated or unwritten logical block, the controller shall operate as described in section 3.3.3.2.1.

Figure 41 shows an example of the relationship between the source LBAs and the destination LBAs.

Figure 41: Source LBA and Destination LBA Relationship Example



3.3.2.1 Fast copy operations

A fast copy operation is a copy operation that uses a method that is expected to be no slower in total elapsed time than the alternative of the host copying the user data by issuing Read commands and Write commands. Unexpected NVM subsystem operating conditions (e.g., nature of concurrent I/O traffic, availability of controller buffer space, errors, and failures) may cause individual copy operations to be slower than host copying of the user data. High performance methods for fast copy operations include non-read/write methods such as copy on write snapshot and copy on write clone and high-bandwidth copies within a tightly integrated NVM subsystem such as an SSD.

A host is able to restrict a Copy command to only perform fast copy operations for user data specified by a Source Range entry by setting the Fast Copy Only (FCO) bit to '1' in that Source Range entry (refer to Figure 39 and Figure 40). If the FCO bit is set to '1' in a Source Range entry and the controller is unable to use fast copy operations to copy the user data specified by that Source Range entry, then the controller shall abort the command with a status code of Fast Copy Not Possible.

If the controller aborts a Copy command with a status code of Fast Copy Not Possible and clears the Do Not Retry (DNR) bit to '0' in the CQE for that command, then the host may retry that Copy command (e.g., by submitting that command to a different controller). If the controller aborts a Copy command with a status code of Fast Copy Not Possible and sets the Do Not Retry (DNR) bit to '1' in the CQE for that command, then the host should not retry that Copy command (e.g., the host may submit Read commands and Write commands to copy the user data).

A controller sets the NVM All Fast Copy (NVMAFC) bit to '1' in the Optional NVM Command Support (ONCS) field of the I/O Command Set Independent Identify Controller data structure (refer to the NVM Express Base Specification) to indicate that for NVM Command Set Copy commands, all copy operations are fast copy operations within the NVM subsystem that contains the controller. If all copy operations within that NVM subsystem are fast copy operations, then the controller should set that bit to '1'.

3.3.2.2 Copy atomicity

If a controller:

- complies with a version of the NVM Command Set Specification later than revision 1.0;
- complies with a version of the NVM Express Base Specification later than revision 2.0; or
- supports the Copy command and either:
    - Source Range Entries Copy Descriptor Format 2h; or
    - Source Range Entries Copy Descriptor Format 3h,

then the controller shall set the NVM Copy Single Atomicity (NVMCSA) bit to '1' in the ONCS field and shall perform the write portion of a Copy command as a single write command to which the atomicity requirements specified in section 2.1.4 apply.

In some situations, these atomicity requirements require the controller to process user data specified by a portion of a Source Range entry as an atomic write operation, or to process user data specified by multiple Source Range entries (and/or portions thereof) as an atomic write operation. For example, consider a controller that is in Multiple Atomicity Mode for the destination namespace with an atomic write size of 8 logical blocks (e.g., as a consequence of the controller setting the NAWUN field to 8h in the Identify Namespace data structure for the NVM Command Set (refer to Figure 114)):

- if that controller processes a Copy command with 3 Source Ranges entries that each consist of 4 logical blocks and a destination LBA range that starts at an atomic boundary, then that controller performs 2 atomic write operations, where the first atomic write operation consists of the 8 logical blocks described by the first two Source Range entries and the second atomic write operation that consists of the 4 logical blocks described by the third Source Range entry); and
- if that controller processes a Copy command with 2 Source Ranges entries that each consist of 16 logical blocks and a destination LBA range that starts at an atomic boundary, then that controller performs 4 atomic write operations, 2 atomic write operations for each Source Range entry where each atomic write operation consists of half of the logical blocks described by a Source Range entry.

A controller is able to limit the implementation impact of these atomicity requirements by reporting appropriate values in the MSRC field, the MSSRL field, and the MCL field (refer to Figure 114).

If the NVMCSA bit in the ONCS field is cleared to '0', then the controller is based on an older version of this specification and the controller:

- always performs the write portion of a Copy command that has a single Source Range entry as a single write command to which the atomicity requirements specified in section 2.1.4 apply; and
- may or may not perform the write portion of a Copy command that has more than one Source Range entry as a separate write command for each Source Range entry. If the write portion of a Copy command is performed as a separate write command for each Source Range entry, then an independent instance of the atomicity requirements in section 2.1.4 applies to copying the user data specified by each Source Range entry.

The value FFFFh for an atomicity parameter specified in section 2.1.4 (refer to Figure 4) indicates that the atomicity of any write command is 10000h logical blocks. For write commands other than the Copy command, that is the largest command size and hence indicates that the command is always atomic.

For a Copy command, 10000h logical blocks is not the largest command size and hence the value FFFFh indicates only that the atomicity of the write portion of the Copy command is 10000h logical blocks. This version of the NVM Command Set standard does not provide any means for a controller to indicate that the write portion of all Copy commands is atomic.

3.3.2.3 Copy within a Single Namespace

This section applies to Copy commands that copy user data within a single namespace and use Source Range Entries Copy Descriptor Format 0h or 1h. Refer to section 3.3.2.4 for Copy commands that use Source Range Entries Copy Descriptor Format 2h or 3h.

If the single namespace that is the source namespace and the destination namespace for a Copy command is formatted with protection information (PI), then the PRINFOR.PRACT bit and the PRINFOW.PRACT bit in the Copy command affect the processing of PI as follows:

- If the PRINFOR.PRACT bit and the PRINFOW.PRACT bit have the same value (i.e., both are set to '1' or both are cleared to '0'), then the controller shall perform the user data copying specified by the Copy command for each Source Range entry with PI processed as specified in section 5.3.2.5.1; and
- If the PRINFOR.PRACT bit and the PRINFOW.PRACT bit have different values (i.e., one bit is set to '1' and one bit is cleared to '0') then the controller shall abort that Copy command with a status code of Invalid Field in Command.

If the single namespace that is the source namespace and the destination namespace for a Copy command is not formatted with PI, then the controller shall ignore the PRINFOR field and the PRINFOW field.

3.3.2.4 Copy across Multiple Namespaces

This section applies to Copy commands that use Source Range Entries Copy Descriptor Format 2h or 3h to copy user data across multiple namespaces and/or within the same namespace. Refer to section 3.3.2.3 for Copy commands that use Source Range Entries Copy Descriptor Formats 0h or 1h.

A controller that supports copying user data across multiple logical block namespaces (i.e., namespaces that use any I/O Command Set that specifies logical blocks) supports any attached logical block namespace as a copy source or a copy destination.

3.3.2.4.1 Matching and Corresponding Formats

Copy command processing imposes restrictions on reformatting of logical block data and metadata that are copied across different namespaces. For all copy source namespaces and the copy destination namespace:

- logical block data size is required to be the same;
- metadata size is required to be the same, except that the copy destination namespace may have a different metadata size if protection information (PI) metadata is inserted or stripped as part of Copy command processing and all metadata is PI metadata; and
- PI type and format settings are required to be the same unless PI is being inserted or stripped as part of Copy command processing.

If the copy source namespaces and the copy destination namespace do not satisfy these restrictions, then the Copy command is unable to copy user data among them. As an alternative, a host is able to use Read commands and Write commands to copy user data via the host.

The specific restrictions are specified in section 3.3.2.4.2, which uses the following terms for formatting restrictions on namespaces specified by a Copy command:

- **matching namespace formats for copy**: requirements that namespace format, PI type, and all other PI parameters for the namespaces be the same.
- **corresponding protection information formats for copy (corresponding PI formats for copy)**: requirements that the namespace formats differ only by the presence or absence of PI.

The specific requirements for each of these terms are specified in the remainder of this section.

Multiple namespaces that are formatted with PI have matching namespace formats for copy if each namespace uses the NVM Command Set or any other command set that both specifies logical blocks and includes the Copy command specified for the NVM Command Set (e.g., the Zoned Namespace Command Set) and all of the namespaces:

- are formatted with the same logical block data size;
- are formatted with the same metadata size; and
- have the same value for each of the following:
    - the End-to-end Data Protection Type Settings (DPS) field in the Identify Namespace data structure (refer to Figure 114);
    - the Protection Information Format Attribute (PIFA) field in the NVM Command Set I/O Command Set specific Identify Namespace data structure (refer to Figure 118);
    - the bits in the Logical Block Storage Tag Mask (LBSTM) field in the NVM Command Set I/O Command Set specific Identify Namespace data structure (refer to Figure 118) that are not ignored by the host; and
    - the following fields in the extended LBA format (refer to Figure 119) that was used to format the namespace;
        - the Protection Information Format (PIF) field;
        - the Qualified Protection Information Format (QPIF) if the PIF field is set to 11b (i.e., Qualified Type); and
        - the Storage Tag Size (STS) field.

Multiple namespaces that are formatted without PI have matching namespace formats for copy if each namespace uses the NVM Command Set or any other I/O Command Set that both specifies logical blocks

and includes the Copy command specified for the NVM Command Set (e.g., the Zoned Namespace Command Set) and all of the namespaces are formatted with both the same logical block data size and the same metadata size.

Multiple namespaces where at least one of the namespaces is formatted with PI and at least one of the namespaces is formatted without PI do not have matching namespace formats for copy.

A namespace that is formatted with PI and a namespace that is formatted without PI have corresponding PI formats for copy if:

- each namespace uses either the NVM Command Set or any other I/O Command Set that both specifies logical blocks and includes the Copy command specified for the NVM Command Set (e.g., the Zoned Namespace Command Set);
- both namespaces are formatted with the same logical block data size;
- the namespace that is formatted with PI is formatted with metadata size equal to PI size (refer to section 5.3.1) (i.e., does not contain any metadata other than PI); and
- the namespace that is formatted without PI is also formatted without any metadata.

A namespace that is formatted with any metadata that is not PI metadata does not have a corresponding PI format for copy with any other namespace.

3.3.2.4.2 Handling of Protection Information

Protection information (PI) and data copying functionality depends on the formats of the source and destination namespaces, the value of the PRINFOR.PRACT bit in the Copy command, and the value of the PRINFOW.PRACT bit in the Copy command as applicable. This functionality is specified for each Source Range entry, as the controller may process Source Range entries concurrently and in any order provided that all Copy command requirements (e.g., atomicity) are satisfied.

If the source namespace for Source Range 0 is formatted without PI, and the destination namespace is formatted without PI, then for each Source Range entry, including Source Range 0:

- if the source namespace for that Source Range entry and the destination namespace have matching namespace formats for copy, then the controller shall perform the user data copying specified by the Copy command for that Source Range entry; and
- if the source namespace for that Source Range entry and the destination namespace do not have matching namespace formats for copy, then the controller:
    - shall not copy any user data specified by that Source Range entry; and
    - shall abort the command with a status code of Incompatible Namespace or Format.

If the source namespace for Source Range 0 is formatted with PI, the destination namespace is formatted with PI, the PRINFOR.PRACT bit is cleared to ‘0’, and the PRINFOW.PRACT bit is cleared to ‘0’, then for each Source Range entry, including Source Range 0:

- if the source namespace for that Source Range entry and the destination namespace have matching namespace formats for copy, then the controller shall perform the user data copying specified by the Copy command for that Source Range entry with PI passed through as specified in section 5.3.2.5.2; and
- if the source namespace for that Source Range entry and the destination namespace do not have matching namespace formats for copy, then the controller:
    - shall not copy any user data specified by that Source Range entry; and

- shall abort the command with a status code of Incompatible Namespace or Format.

If the source namespace for Source Range 0 is formatted with PI, the destination namespace is formatted with PI, the PRINFOR.PRACT bit is set to '1' and the PRINFOW.PRACT bit is set to '1', then for each Source Range entry, including Source Range 0:

- if the source namespace for that Source Range entry and the destination namespace have matching namespace formats for copy, then the controller shall perform the user data copying specified by the Copy command for that Source Range entry with PI replaced as specified in section 5.3.2.5.2; and
- if the source namespace for that Source Range entry and the destination namespace do not have matching namespace formats for copy, then the controller:
  - shall not copy any user data specified by that Source Range entry; and
  - shall abort the command with a status code of Incompatible Namespace or Format.

If the source namespace for Source Range 0 is formatted with PI, the destination namespace is formatted with PI, and the PRINFOR.PRACT bit has a different value from the PRINFOW.PRACT bit, then the controller shall abort the command with a status code of Invalid Field in Command.

If the source namespace for Source Range 0 is formatted without PI, the destination namespace is formatted with PI, and the PRINFOW.PRACT bit is set to '1', then for each Source Range entry, including Source Range 0:

- if the source namespace for that Source Range entry is formatted without PI and has a corresponding PI format for copy with the destination namespace, then the controller shall perform the user data copying specified by the Copy command for that Source Range entry with PI inserted as specified in section 5.3.2.5.2; and
- if the source namespace for that Source Range entry is formatted with PI or does not have a corresponding PI format for copy with the destination namespace, then the controller:
  - shall not copy any user data specified by that Source Range entry; and
  - shall abort the command with a status code of Incompatible Namespace or Format.

If the source namespace for Source Range 0 is formatted without PI, the destination namespace is formatted with PI, and the PRINFOW.PRACT bit is cleared to '0', then the controller shall abort the command with a status code of Incompatible Namespace or Format.

If the source namespace for Source Range 0 is formatted with PI, the destination namespace is formatted without PI, and the PRINFOR.PRACT bit is set to '1', then for each Source Range entry, including Source Range 0:

- if the source namespace for that Source Range entry is formatted with PI and has a corresponding PI format for copy with the destination namespace, then the controller shall perform the data copying specified by the Copy command for that Source Range entry with PI stripped as specified in section 5.3.2.5.2; and
- if the source namespace for that Source Range entry is not formatted with PI or does not have a corresponding PI format for copy with the destination namespace, then the controller:
  - shall not copy any user data specified by that Source Range entry; and
  - shall abort the command with a status code of Incompatible Namespace or Format.

If the source namespace for Source Range 0 is formatted with PI, the destination namespace is formatted without PI, and the PRINFOR.PRACT bit is cleared to '0', then the controller shall abort the command with a status code of Incompatible Namespace or Format.

3.3.2.5 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.

If the command completes with failure (i.e., completes with a status code other than Successful Completion), then:

- the controller may or may not have copied some of the user data;
- Dword 0 of the completion queue entry contains the number of the lowest numbered Source Range entry that was not successfully copied (e.g., if Source Range 0, Source Range 1, Source Range 2, and Source Range 5 are copied successfully and Source Range 3 and Source Range 4 are not copied successfully, then Dword 0 is set to 3); and
- prior to aborting the command, the controller may or may not have copied some (or all) of the data specified by that Copy command for the Source Range entries that have a number greater than or equal to the value in Dword 0.

If no data was written to the destination LBAs, then Dword 0 of the completion queue entry shall be cleared to 0h.

Copy command specific errors are defined in Figure 42.

Figure 42: Copy – Command Specific Status Values

Value	Definition
81h	**Invalid Protection Information:** The protection information specified by the command is invalid due to: • The Protection Information Read (PRINFOR) field or Protection Information Write (PRINFOW) field (refer to Figure 34) containing an invalid value for the Protection Information with which the namespace was formatted (refer to the PI field in the Format NVM Command section in the NVM Express Base Specification and the DPS field in Figure 114); • the ILBRT field being invalid (refer to section 5.3.3); or • the EILBRT field in a Source Range entry being invalid (refer to section 5.3.3).
82h	**Attempted Write to Read Only Range:** The destination LBA range specified contains read-only blocks. The controller shall not return this status value if the read-only condition on the media is a result of a change in the write protection state of a namespace (refer to the Namespace Write Protection section in the NVM Express Base Specification).
83h	**Command Size Limit Exceeded:** One or more of the Copy command processing limits (i.e., non-zero value of the NR, MSSRL, and MCL fields in the Identify Namespace data structure) was exceeded.
85h	**Incompatible Namespace or Format:** At least one source namespace and the destination namespace have incompatible formats (refer to section 3.3.2.4).
86h	**Fast Copy Not Possible:** The Fast Copy Only (FCO) bit was set to '1' in a Source Range entry and the controller was not able to use fast copy operations to copy the specified data (refer to section 3.3.2.1).
87h	**Overlapping I/O Range:** A source logical block range overlaps the destination logical block range (refer to section 3.3.2).
89h	**Insufficient Resources:** A resource shortage prevented the controller from performing the requested copy. The host should not retry the command on the same controller.
8Ah	**Namespace Not Reachable:** One or more of the Source Range entries specifies the NSID of a namespace that is not contained in any Reachability Group that is in a Reachability Association with the Reachability Group that contains the destination namespace. Refer to the Reachability Reporting architecture section in the NVM Express Base Specification.

3.3.3 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 43: Dataset Management – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the data to use for the command. Refer to the Common Command Format figure in the NVM Express Base Specification for the definition of this field.

Figure 44: Dataset Management – Command Dword 10

Bits	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 45: Dataset Management – Command Dword 11

Bits	Description
31:03	Reserved
02	**Attribute – Deallocate (AD):** If this bit is set to '1', then the NVM subsystem may deallocate all provided ranges. The data returned for logical blocks that were deallocated is specified in section 3.3.3.2.1. The data and metadata for logical blocks that are not deallocated by the NVM subsystem are not changed as the result of a Dataset Management command.
01	**Attribute – Integral Dataset for Write (IDW):** If this bit is 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 this bit is 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 45 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 Length in Logical Blocks field is a 1-based value. The definition of the Dataset Management command Range field is specified in Figure 46. The maximum case of 256 ranges is shown.

Figure 46: Dataset Management – Range Definition

Range	Bytes	Field
Range 0	03:00	**Context Attributes (CATTR)**
	07:04	**Length in Logical Blocks (LLB)**
	15:08	**Starting LBA (SLBA)**
Range 1	19:16	**Context Attributes (CATTR)**
	23:20	**Length in Logical Blocks (LLB)**
	31:24	**Starting LBA (SLBA)**
…		
Range 255	4083:4080	**Context Attributes (CATTR)**
	4087:4084	**Length in Logical Blocks (LLB)**
	4095:4088	**Starting LBA (SLBA)**

3.3.3.1 Dataset Management Processing Limits

Processing limits for Dataset Management commands are indicated by non-zero values in three fields in the Identify Controller data structure (refer to the Identify Controller data structure section in the NVM Express Base Specification):

a) A non-zero value in the Dataset Management Ranges Limit (DMRL) field indicates a processing limit on the number of ranges. If the controller reports a non-zero value in this field, then the controller does not process attributes and context attributes for any range whose range number (i.e., the number of the range as specified in the Range column of Figure 46) is greater than or equal to that non-zero value.
b) A non-zero value in the Dataset Management Range Size Limit (DMRSL) field indicates a processing limit on the number of logical blocks in a single range. If the controller reports a non-zero value in this field, then the controller does not process attributes and context attributes for any logical block whose LBA offset from the starting LBA of the range (refer to Figure 46) that specifies the logical block is greater than or equal to that non-zero value.
c) A non-zero value in the Dataset Management Size Limit (DMSL) field indicates a processing limit on the total number of logical blocks for the command. If the controller reports a non-zero value in this field, then the controller does not process attributes and context attributes for any logical block specified by any range for which the sum of:

    a. the number of logical blocks, specified by lower numbered ranges, if any, otherwise zero; and
    b. the LBA offset for that logical block from the starting LBA of that range,

    is greater than or equal to that non-zero value.

A logical block specified by a Dataset Management command satisfies all three of these processing limits if and only if each processing limit does not prevent controller processing of attributes and context attributes for that logical block. A logical block specified by a Dataset Management command does not satisfy a processing limit if that limit prevents controller processing of attributes and context attributes for that logical block.

The controller shall set all three processing limit fields (i.e., the DMRL, DMRSL and DMSL fields) to non-zero values or shall clear all three processing limit fields to 0h. A controller is able to impose a subset of the three processing limits by setting the field that reports each unused processing limit to the maximum possible value for that field (i.e., all bits set to ‘1’), with the exception that the resulting processing limit for the number of ranges is 255 of the 256 ranges supported by the Dataset Management command. Note that

this exception is due to the DMRL field being 1-based in contrast to the 0's-based Number of Ranges (NR) field in the Dataset Management command.

If all three processing limit fields (i.e., the DMRL, DMRSL and DMSL fields) contain non-zero values, then the controller supports the Dataset Management command and:

a) Each processing limit field indicates a processing limit for controller processing of attributes and context attributes for logical blocks specified by the command;
b) If Dataset Management Support Variants (NVMDSMSV) bit is set to '1' in the Optional NVM Command Support (ONCS) field in the Identify Controller data structure, then for the logical blocks specified by the command:
    a. The controller should process attributes and context attributes for all logical blocks that satisfy all three processing limits; and
    b. The controller should not process attributes and context attributes for any logical blocks that do not satisfy one or more of the three processing limits;

    and

c) If the NVMDSMSV bit is cleared to '0', then for the logical blocks specified by the command:
    a. If all logical blocks specified by the command satisfy all three processing limits, then the controller shall process attributes and context attributes for those logical blocks; and
    b. If the command specifies any logical block that does not satisfy one or more of the three processing limits, then the controller shall abort the command with Command Size Limit Exceeded status.

If all three processing limit fields (i.e., the DMRL, DMRSL and DMSL fields) are cleared to 0h then:

a) If the NVMDSMSV bit is set to '1', then the controller supports the Dataset Management command and does not report any processing limits on the number of ranges, number of logical blocks in a single range or total number of logical blocks for the command; and
b) If the NVMDSMSV bit is cleared to '0', then the controller does not support the Dataset Management command.

A controller may choose to take no action on any or all logical blocks for which attributes or context attributes are processed. If a Dataset Management command contains one or more ranges for which neither attributes nor context attributes are processed, then a controller may nonetheless check the fields that specify such ranges and abort the command if an error is detected (e.g., if the controller detects that such a range extends beyond the size of the namespace).

3.3.3.1.1 Dataset Management Processing Limits Example

For example, under the assumptions that the Dataset Management Support Variants (NVMDSMSV) bit is set to '1' in the ONCS field and the DMRSL field is set to its maximum value, consider a Dataset Management command that specifies two ranges, with range 0 containing 1,024 logical blocks and range 1 containing 512 logical blocks:

a) if the DMRL field is set to 1 and the DMSL field is set to 1,048, then the controller is expected to process attributes and context attributes for the logical blocks specified by range 0, and the controller does not process either attributes or context attributes for the logical blocks contained in range 1; and
b) if the DMRL field is set to 2 and the DMSL field is set to 1,048, then the controller is expected to process attributes and context attributes for the logical blocks specified by range 0 and for the first 24 logical blocks of range 1, and the controller does not process either attributes or context attributes for the other logical blocks (i.e., 25 - 512) contained in range 1.

3.3.3.2 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 47: Dataset Management – Context Attributes

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:24</td><td>Command Access Size (CASZE): This field is the 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>23:11</td><td>Reserved</td></tr>
<tr><td>10</td><td>Write Prepare (WPREP): If this bit is set to ‘1’, then the provided range is expected to be written in the near future.</td></tr>
<tr><td>09</td><td>Sequential Write Range (SWR): If this bit is 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>08</td><td>Sequential Read Range (SRR): If this bit is 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>07:06</td><td>Reserved</td></tr>
<tr><td>05:04</td><td>Access Latency (AL): This field specifies the expected access latency.
<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 (AF): This field specifies the expected access frequency.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0h</td><td>No frequency information provided.</td></tr>
<tr><td>1h</td><td>Typical number of reads and writes expected for this LBA range.</td></tr>
<tr><td>2h</td><td>Infrequent writes and infrequent reads to the LBA range indicated.</td></tr>
<tr><td>3h</td><td>Infrequent writes and frequent reads to the LBA range indicated.</td></tr>
<tr><td>4h</td><td>Frequent writes and infrequent reads to the LBA range indicated.</td></tr>
<tr><td>5h</td><td>Frequent writes and frequent reads to the LBA range indicated.</td></tr>
<tr><td>6h to Fh</td><td>Reserved</td></tr>
</table></td></tr>
</table>

3.3.3.2.1 Deallocated or Unwritten Logical Blocks

A logical block that has never been written to, or which has been deallocated using the Dataset Management command, the Write Zeroes command or the Sanitize command is called a deallocated or unwritten logical block.

Using the Error Recovery feature (refer to section 4.1.3.2), host software may select the behavior of the controller when reading deallocated or unwritten blocks. The controller shall abort Copy, Read, Verify, or Compare commands that include deallocated or unwritten blocks with a status of Deallocated or Unwritten Logical Block if that error has been enabled using the DULBE bit in the Error Recovery feature. If the Deallocated or Unwritten Logical error is not enabled, the values read from a deallocated or unwritten block and its metadata (excluding protection information) shall be:

- all bytes cleared to 0h if the Deallocation Read Behavior (DRB) field in the DLFEAT field is set to 001b;
- all bytes set to FFh if the DRB field is set to 010b; or

- either all bytes cleared to 0h or all bytes set to FFh if the DRB field is cleared to 000b.

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 operation occurs to that logical block. A deallocated or unwritten block is no longer deallocated or unwritten when the logical block is written. Read operations and Verify operations do not affect the deallocation status of a logical block.

The values read from a deallocated or unwritten logical block's protection information field shall:

- have each byte in the Guard field value set to FFh or set to the CRC for the value read from the deallocated logical block and its metadata (excluding protection information) (e.g., cleared to 0h if the value read is all bytes cleared to 0h); and
- have each byte in the Storage Tag field (if defined), the Application Tag field, and the Logical Block Reference Tag field set to FFh (indicating the protection information shall not be checked).

Using the Error Recovery feature (refer to section 4.1.3.5), host software may enable an error to be returned if a deallocated or unwritten logical block is read. If this error is supported for the namespace and enabled, then any User Data Read Access Command that includes a deallocated or unwritten logical block shall abort with the Deallocated or Unwritten 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-5 and SCSI UNMAP command described in SBC-4.

3.3.3.3 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 (i.e., SCT field set to 1h) are shown in Figure 48.

Figure 48: Dataset Management – Command Specific Status Values

Value	Definition
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. The controller shall not return this status value if the read-only condition on the media is a result of a change in the write protection state of a namespace (refer to the Namespace Write Protection section in the NVM Express Base Specification).
83h	**Command Size Limit Exceeded:** One or more of the Dataset Management processing limits (i.e., non-zero values of the DMRL, DMRSL and DMSL fields in the Identify Controller data structure) was exceeded (refer to section 3.3.3.1). The controller shall not return this status value if the Dataset Management Support Variants (NVMDSMSV) bit is set to '1' in the Optional NVM Command Support field in the Identify Controller data structure.

3.3.4 Read command

The Read command reads data and metadata, if applicable, from the I/O 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 2, Command Dword 3, 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 49: Read – Metadata Pointer

Bits	Description
63:00	**Metadata Pointer (MPTR):** This field contains the Metadata Pointer, if applicable. Refer to the Common Command Format figure in the NVM Express Base Specification for the definition of this field.

Figure 50: Read – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies where data is transferred to. Refer to the Common Command Format figure in the NVM Express Base Specification for the definition of this field.

Figure 51: Read – Command Dword 2 and Dword 3

Bits	Description
63:48	Reserved
47:00	**Expected Logical Block Tags Upper (ELBTU):** This field and Command Dword 14 specify the variable sized Expected Logical Block Storage Tag (ELBST) and Expected Initial Logical Block Reference Tag (EILBRT) fields, which are defined in section 5.3.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller.

Figure 52: Read – Command Dword 10 and Command Dword 11

Bits	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 53: Read – Command Dword 12

Bits	Description
31	**Limited Retry (LR):** If this bit is set to '1', then the controller should apply limited retry efforts. If this bit is cleared to '0', then the controller should apply all available error recovery means to return the data to the host.
30	**Force Unit Access (FUA):** If this bit is set to '1', then for data and metadata, if any, associated with logical blocks specified by the Read command, the controller shall: 1) commit that data and metadata, if any, to non-volatile medium; and 2) return the data, and metadata, if any, that are read from non-volatile medium. There is no implied ordering with other commands. If this bit is cleared to '0', then this bit has no effect.
29:26	**Protection Information (PRINFO):** Specifies the protection information action and check field, as defined in Figure 11.
25	Reserved
24	**Storage Tag Check (STC):** This bit specifies the Storage Tag field shall be checked as part of end-to-end data protection processing as defined in Figure 12.
23:20	Reserved
19:16	**Command Extension Type (CETYPE):** Specifies the Command Extension Type that applies to the command (refer to the Key Per I/O section in the NVM Express Base Specification).
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.

The definition of Command Dword 13 is based on the CETYPE value. If the CETYPE value is cleared to 0h, then Command Dword 13 is defined in Figure 54. If the CETYPE value is non-zero, then Command Dword 13 is defined in Figure 55.

Figure 54: Read – Command Dword 13 if CETYPE is cleared to 0h

<table>
<tr><th>Bits</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>Description</th></tr>
<tr><td>07</td><td>Incompressible (INCPRS): If this bit is set to ‘1’, then data is not compressible for the logical blocks indicated. If this bit is cleared to ‘0’, then no information on compression is provided.</td></tr>
<tr><td>06</td><td>Sequential Request (SEQREQ): If this bit is set to ‘1’, then this command is part of a sequential read that includes multiple Read commands. If this bit is cleared to ‘0’, then no information on sequential access is provided.</td></tr>
<tr><td>05:04</td><td>Access Latency (AL): This field specifies the expected access latency.
<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 (AF): This field specifies the expected access frequency.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0h</td><td>No frequency information provided.</td></tr>
<tr><td>1h</td><td>Typical number of reads and writes expected for this LBA range.</td></tr>
<tr><td>2h</td><td>Infrequent writes and infrequent reads to the LBA range indicated.</td></tr>
<tr><td>3h</td><td>Infrequent writes and frequent reads to the LBA range indicated.</td></tr>
<tr><td>4h</td><td>Frequent writes and infrequent reads to the LBA range indicated.</td></tr>
<tr><td>5h</td><td>Frequent writes and frequent reads to the LBA range indicated.</td></tr>
<tr><td>6h</td><td>One time read. E.g., command is due to virus scan, backup, file copy, or archive.</td></tr>
<tr><td>7h</td><td>Speculative read. The command is part of a prefetch operation.</td></tr>
<tr><td>8h</td><td>The LBA range is going to be overwritten in the near future.</td></tr>
<tr><td>9h to Fh</td><td>Reserved</td></tr>
</table></td></tr>
</table></td></tr>
</table>

Figure 55: Read - Command Dword 13 if CETYPE is non-zero

Bits	Description
31:16	Reserved
15:00	**Command Extension Value (CEV):** The definition of this field is dependent on the value of the CETYPE field. Refer to the Key Per I/O section in the NVM Express Base Specification.

Figure 56: Read – Command Dword 14

Bits	Description
31:00	**Expected Logical Block Tags Lower (ELBTL):** This field and bits 47:00 of Command Dword 2 and Dword 3 specify the variable sized Expected Logical Block Storage Tag (ELBST) and Expected Initial Logical Block Reference Tag (EILBRT) fields, which are defined in section 5.3.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller.

Figure 57: Read – Command Dword 15

Bits	Description
31:16	**Expected Logical Block Application Tag Mask (ELBATM):** This field specifies the Application Tag Mask expected value. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.
15:00	**Expected Logical Block Application Tag (ELBAT):** This field specifies the Application Tag expected value. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.

3.3.4.1 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 58.

Figure 58: 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 (PRINFO) field (refer to Figure 53) settings specified in the command are invalid for the Protection Information with which the namespace was formatted (refer to the PI field in Figure 90 and the DPS field in Figure 114) or the EILBRT field is invalid (refer to section 5.3.3).

3.3.5 Verify command

The Verify command verifies integrity of stored information by reading data and metadata, if applicable, for the LBAs indicated without transferring any data or metadata to the host. A Verify operation consists of the controller actions (e.g., reading) that verify integrity of stored information during execution of a Verify command. The command may specify protection information to be checked as part of the Verify operation.

Verify operations may be implemented via integrity checks of stored data and metadata. Metadata integrity checks shall include protection information if the Verify command specifies checking of protection information and the namespace is formatted with protection information.

If reading the data and metadata, if applicable, would result in an error being returned, then an error shall be returned as a result of the Verify operation on that data and metadata, if applicable. In this situation, the error that results from integrity checks may differ from the error that would result from reading (e.g., there is no requirement that the Verify and Read commands return the same error). Setting the Limited Retry (LR) bit to ‘1’ shall have the same effect in both the Read and Verify commands.

All data that is read or has its integrity checked by a Verify operation shall be included in the value of the Data Units Read field in the SMART/Health Information log page, refer to the SMART / Health Information section in the NVM Express Base Specification.

If the Verify Size Limit (VSL) field in the Identify Controller data structure is set to a non-zero value and:

a) if the Verify Support (NVMVFYS) bit in the Optional NVM Command Support field in the Identify Controller data structure is set to ‘1’, then the VSL field indicates the recommended maximum data size for the Verify command and any Verify command that specifies a logical block range whose data size exceeds that recommended maximum may encounter delays in processing; and
b) if the NVMVFYS bit is cleared to ‘0’, then the VSL field indicates the data size limit for the Verify command, and the controller shall abort any Verify command that specifies a logical block range whose data size exceeds that limit with a status code of Invalid Field in Command.

The command uses Command Dword 2, Command Dword 3, Command Dword 10, Command Dword 11, Command Dword 12, Command Dword 13, Command Dword 14, and Command Dword 15 fields.

Figure 59: Verify – Command Dword 2 and Dword 3

Bits	Description
63:48	Reserved
47:00	**Expected Logical Block Tags Upper (ELBTU):** This field and Command Dword 14 specify the variable sized Expected Logical Block Storage Tag (ELBST) and Expected Initial Logical Block Reference Tag (EILBRT) fields, which are defined in section 5.3.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller.

Figure 60: Verify – Command Dword 10 and Command Dword 11

Bits	Description
63:00	**Starting LBA (SLBA):** This field indicates the 64-bit address of the first logical block of data to be verified as part of the operation. Command Dword 10 contains bits 31:00; Command Dword 11 contains bits 63:32.

Figure 61: Verify – Command Dword 12

Bits	Description
31	**Limited Retry (LR):** If this bit is set to '1', then the controller should apply limited retry efforts. If this bit is cleared to '0', then the controller should apply all available error recovery means before completing the command with failure.
30	**Force Unit Access (FUA):** If this bit is set to '1', then the controller shall flush any data and metadata specified by the Verify command from any volatile cache before performing the Verify operation and shall perform the Verify operation on data and metadata that have been committed to non-volatile medium. There is no implied ordering with other commands. If this bit is cleared to '0', then this bit has no effect.
29:26	**Protection Information (PRINFO):** Specifies the protection information action and check field, as defined in Figure 11. The Protection Information Check (PRCHK) field in the PRINFO field specifies the protection information to be checked by the Verify operation. The Protection Information Action (PRACT) bit in the PRINFO field is cleared to '0' by the host. If the PRACT bit is not cleared to '0', then the controller shall abort the command with a status of Invalid Field in Command.
25	Reserved
24	**Storage Tag Check (STC):** This bit specifies the Storage Tag field shall be checked as part of Verify operation as defined in Figure 12.
23:20	Reserved
19:16	**Command Extension Type (CETYPE):** Specifies the Command Extension Type that applies to the command (refer to the Key Per I/O section in the NVM Express Base Specification).
15:00	**Number of Logical Blocks (NLB):** This field indicates the number of logical blocks to be verified. This is a 0's based value.

The definition of Command Dword 13 is based on the CETYPE value. If the CETYPE value is cleared to 0h, then Command Dword 13 is reserved. If the CETYPE value is non-zero, then Command Dword 13 is defined in Figure 62.

Figure 62: Verify - Command Dword 13 if CETYPE is non-zero

Bits	Description
31:16	Reserved
15:00	**Command Extention Value (CEV):** The definition of this field is dependent on the value of the CETYPE field. Refer to the Key Per I/O section in the NVM Express Base Specification.

Figure 63: Verify – Command Dword 14

Bits	Description
31:00	**Expected Logical Block Tags Lower (ELBTL):** This field and bits 47:00 of Command Dword 2 and Dword 3 specify the variable sized Expected Logical Block Storage Tag (ELBST) and Expected Initial Logical Block Reference Tag (EILBRT) fields, which are defined in section 5.3.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller.

Figure 64: Verify – Command Dword 15

Bits	Description
31:16	**Expected Logical Block Application Tag Mask (ELBATM):** This field specifies the Application Tag Mask expected value. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.
15:00	**Expected Logical Block Application Tag (ELBAT):** This field specifies the Application Tag expected value. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.

3.3.5.1 Command Completion

Upon completion of the Verify command, the controller posts a completion queue entry (CQE) to the associated I/O Completion Queue. The status code types and values that may be used in a CQE for the Verify command include the status code type and status code values for all Media and Data Integrity Errors for the NVM Command Set that are applicable to the Read command (e.g., Unrecovered Read Error). For more information of status codes for the NVM Command Set refer to section 3.1.

Verify command specific status values are defined in Figure 65.

Figure 65: Verify – Command Specific Status Values

Value	Description
81h	**Invalid Protection Information:** The Protection Information (PRINFO) field (refer to Figure 61) settings specified in the command are invalid for the Protection Information with which the namespace was formatted (refer to the PI field in Figure 90 and the DPS field in Figure 114) or the EILBRT field is invalid (refer to section 5.3.3).

3.3.6 Write command

The Write command writes data and metadata, if applicable, to the I/O 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 2, Command Dword 3, 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 66: Write – Metadata Pointer

Bits	Description
63:00	**Metadata Pointer (MPTR):** This field contains the Metadata Pointer, if applicable. Refer to the Common Command Format figure in the NVM Express Base Specification for the definition of this field.

Figure 67: Write – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the location of a data buffer where data is transferred from. Refer to the Common Command Format figure in the NVM Express Base Specification for the definition of this field.

Figure 68: Write – Command Dword 2 and Dword 3

Bits	Description
63:48	Reserved
47:00	**Logical Block Tags Upper (LBTU):** This field and Command Dword 14 specify the variable sized Logical Block Storage Tag (LBST) and Initial Logical Block Reference Tag (ILBRT) fields, which are defined in section 5.3.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller.

Figure 69: Write – Command Dword 10 and Command Dword 11

Bits	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 70: Write – Command Dword 12

Bits	Description
31	**Limited Retry (LR):** If this bit is set to '1', then the controller should apply limited retry efforts. If this bit is cleared to '0', then the controller should apply all available error recovery means to write the data to the NVM.
30	**Force Unit Access (FUA):** If this bit is set to '1', then for data and metadata, if any, associated with logical blocks specified by the Write command, the controller shall write that data and metadata, if any, to non-volatile medium before indicating command completion. There is no implied ordering with other commands. If this bit is cleared to '0', then this bit has no effect.
29:26	**Protection Information (PRINFO):** Specifies the protection information action and check field, as defined in Figure 11.
25	Reserved
24	**Storage Tag Check (STC):** This bit specifies the Storage Tag field shall be checked as part of end-to-end data protection processing as defined in Figure 12.
23:20	**Directive Type (DTYPE)**: Specifies the Directive Type associated with the Directive Specific field (refer to the Directives section in the NVM Express Base Specification).
19:16	**Command Extension Type (CETYPE):** Specifies the Command Extension Type that applies to the command (refer to the Key Per I/O section in the NVM Express Base Specification).
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.

The definition of Command Dword 13 is based on the CETYPE value. If the CETYPE value is cleared to 0h, then Command Dword 13 is defined in Figure 71. If the CETYPE value is non-zero, then Command Dword 13 is defined in Figure 72.

Figure 71: Write – Command Dword 13 if CETYPE is cleared to 0h

Bits	Description
31:16	**Directive Specific (DSPEC)**: Specifies the Directive Specific value associated with the Directive Type field (refer to the Directives section in the NVM Express Base Specification).
15:08	Reserved

Figure 71: Write – Command Dword 13 if CETYPE is cleared to 0h

<table>
<tr><th>Bits</th><th>Description</th></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>Definition</th></tr>
<tr><td>07</td><td>Incompressible (INCPRS): If this bit is set to ‘1’, then data is not compressible for the logical blocks indicated. If this bit is cleared to ‘0’, then no information on compression is provided.</td></tr>
<tr><td>06</td><td>Sequential Request (SEQREQ): If this bit is set to ‘1’, then this command is part of a sequential write that includes multiple Write commands. If this bit is cleared to ‘0’, then no information on sequential access is provided.</td></tr>
<tr><td>05:04</td><td>Access Latency (AL): This field specifies the exected access latency.
<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 (AF): This field specifies the expected access frequency.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0h</td><td>No frequency information provided.</td></tr>
<tr><td>1h</td><td>Typical number of reads and writes expected for this LBA range.</td></tr>
<tr><td>2h</td><td>Infrequent writes and infrequent reads to the LBA range indicated.</td></tr>
<tr><td>3h</td><td>Infrequent writes and frequent reads to the LBA range indicated.</td></tr>
<tr><td>4h</td><td>Frequent writes and infrequent reads to the LBA range indicated.</td></tr>
<tr><td>5h</td><td>Frequent writes and frequent reads to the LBA range indicated.</td></tr>
<tr><td>6h</td><td>One time write. E.g., command is due to virus scan, backup, file copy, or archive.</td></tr>
<tr><td>7h to Fh</td><td>Reserved</td></tr>
</table></td></tr>
</table></td></tr>
</table>

Figure 72: Write - Command Dword 13 if CETYPE is non-zero

Bits	Description
31:16	**Directive Specific (DSPEC):** Specifies the Directive Specific value associated with the Directive Type field (refer to the Key Per I/O section in the NVM Express Base Specification).
15:00	**Command Extension Value (CEV):** The definition of this field is dependent on the value of the CETYPE field. Refer to the Key Per I/O section in the NVM Express Base Specification.

Figure 73: Write – Command Dword 14

Bits	Description
31:00	**Logical Block Tags Lower (LBTL):** This field and bits 47:00 of Command Dword 2 and Dword 3 specify the variable sized Logical Block Storage Tag (LBST) and Initial Logical Block Reference Tag (ILBRT) fields, which are defined in section 5.3.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller.

Figure 74: Write – Command Dword 15

Bits	Description
31:16	**Logical Block Application Tag Mask (LBATM):** This field specifies the Application Tag Mask value. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.
15:00	**Logical Block Application Tag (LBAT):** This field specifies the Application Tag value. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.

3.3.6.1 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 (i.e., SCT fields set to 1h) are shown in Figure 75.

Figure 75: Write – Command Specific Status Values

Value	Definition
80h	**Conflicting Attributes:** The attributes specified in the command are conflicting.
81h	**Invalid Protection Information:** The Protection Information (PRINFO) field (refer to Figure 70) settings specified in the command are invalid for the Protection Information with which the namespace was formatted (refer to the PI field in Figure 90 and the DPS field in Figure 114) or the ILBRT field is invalid (refer to section 5.3.3).
82h	**Attempted Write to Read Only Range:** The LBA range specified contains read-only blocks. The controller shall not return this status value if the read-only condition on the media is a result of a change in the write protection state of a namespace (refer to the Namespace Write Protection section in the NVM Express Base Specification).

3.3.7 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.

If the Write Uncorrectable Size Limit (WUSL) field in the Identify Controller data structure is set to a non-zero value and:

a) if the Write Uncorrectable Support Variants (NVMWUSV) bit is set to '1' in the Optional NVM Command Support field in the Identify Controller data structure, then the WUSL field indicates the recommended maximum data size for the Write Uncorrectable command and any Write Uncorrectable command that specifies a logical block range whose data size exceeds that recommended maximum may encounter delays in processing; and
b) if the NVMWUSV bit is cleared to '0', then the WUSL field indicates the data size limit for the Write Uncorrectable command, and the controller shall abort any Write Uncorrectable command that specifies a logical block range whose data size exceeds that limit with a status of Invalid Field in Command.

The fields used are Command Dword 10, Command Dword 11, and Command Dword 12 fields. All other command specific fields are reserved.

Figure 76: Write Uncorrectable – Command Dword 10 and Command Dword 11

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

Figure 77: Write Uncorrectable – Command Dword 12

Bits	Description
31:24	Reserved
23:20	**Directive Type (DTYPE):** Specifies the Directive Type associated with the Directive Specific field (refer to the Directives section in the NVM Express Base Specification).
19:16	Reserved

Figure 77: Write Uncorrectable – Command Dword 12

Bits	Description
15:00	**Number of Logical Blocks (NLB):** This field specifies the number of logical blocks to become uncorrectable. This is a 0's based value.

Figure 78: Write Uncorrectable – Command Dword 13

Bits	Description
31:16	**Directive Specific (DSPEC):** Specifies the Directive Specific value associated with the Directive Type field (refer to the Directives section in the NVM Express Base Specification).
15:00	Reserved

3.3.7.1 Command Completion

Upon command completion, the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command.

Write Uncorrectable command specific errors (i.e., SCT field set to 1h) are shown in Figure 79.

Figure 79: Write Uncorrectable – Command Specific Status Values

Value	Description
82h	**Attempted Write to Read Only Range:** The LBA range specified contains read-only blocks. The controller shall not return this status value if the read-only condition on the media is a result of a change in the write protection state of a namespace (refer to the Namespace Write Protection section in the NVM Express Base Specification).

3.3.8 Write Zeroes command

The Write Zeroes command is used to clear a range of logical blocks or all of the logical blocks in an entire namespace to zero. Non-PI related metadata for this command, if any, shall be all bytes cleared to 0h. The protection information for logical blocks written to the media is updated based on CDW12.PRINFO. If the Protection Information Action (PRACT) bit is cleared to '0', then the protection information for this command shall be all zeroes. If the PRACT bit is set to '1', then the protection information shall be based on the End-to-end Data Protection Type Settings (DPS) field in the Identify Namespace data structure (refer to Figure 114), CDW15.LBATM, CDW15.LBAT, as well as CDW2/3 and CDW14 content as described in section 5.3.1.4.1. Protection information of all zeroes is generated if the PRACT bit is cleared to 0h resulting in invalid protection information; therefore, the host should set the PRACT bit to '1' to generate valid protection information.

After successful completion of a Write Zeroes command that specifies:

- clearing a range of logical blocks to zero, the value returned by subsequent successful reads of logical blocks and associated metadata (excluding protection information) in this range shall be all bytes cleared to 0h until a write occurs to that range of logical blocks; or
- clearing all of the logical blocks in an entire namespace to zero, the value returned by subsequent reads of each logical block in that namespace shall be all bytes cleared to 0h until a write occurs to that logical block.

For each logical block in the range specified by a Write Zeroes command, if the namespace supports clearing all bytes to 0h in the values read (e.g., the Deallocation Read Behavior (DRB) field in the DLFEAT

field is set to 001b) from a deallocated logical block and its metadata (excluding protection information), and the value of the Deallocate bit (CDW12.DEAC) in that Write Zeroes command is:

- set to '1', then the controller should deallocate that logical block; and
- cleared to '0', then the controller may deallocate that logical block.

For each logical block in the range specified by a Write Zeroes command, if the namespace does not support clearing all bytes to 0h in the values read from that logical block and its metadata (excluding the protection information) when that logical block is deallocated, then the controller shall not deallocate that logical block.

If a logical block in the range specified by a Write Zeroes command is deallocated as a result of that command, then:

- the DULBE bit in the Error Recovery feature (refer to section 4.1.3.2) affects whether subsequent reads of that deallocated logical block are able to succeed (refer to section 3.3.3.2.1); and
- the values of protection information, if any, returned by subsequent successful reads of that deallocated logical block are specified in section 3.3.3.2.1.

If a logical block in the range specified by a Write Zeroes command is not deallocated as a result of that command, then the values of protection information, if any, returned by subsequent successful reads of that logical block shall be based on CDW12.PRINFO in that Write Zeroes command.

If the Write Zeroes Size Limit (WZSL) field in the I/O Command Set specific Identify Controller data structure for the NVM Command Set (refer to Figure 120) is set to a non-zero value and the Write Zeroes with Deallocate Size Limit (WZDSL) field in that data structure is cleared to 0h, then:

a) if the Write Zeroes Support Variants (NVMWZSV) bit is set to '1' in the Optional NVMe Command Support field in the Identify Controller data structure, then the WZSL field indicates the recommended maximum data size for the Write Zeroes command and any Write Zeroes command that specifies a logical block range whose data size exceeds that recommended maximum may encounter delays in processing; and
b) if the NVMWZSV bit is cleared to '0', then the WZSL field indicates the maximum data size limit for the Write Zeroes command, and the controller shall abort any Write Zeroes command that specifies a logical block range whose data size exceeds that limit with a status of Invalid Field in Command.

If the WZSL field is set to a non-zero value and the WZDSL field is set to a non-zero value, then:

a) if the NVMWZSV bit is set to '1', then:
    - the WZDSL field indicates the recommended maximum data size for any Write Zeroes command that has the Deallocate bit set to '1';
    - the WZSL field indicates the recommended maximum data size for any Write Zeroes command that has the Deallocate bit cleared to '0'; and
    - any Write Zeroes command that specifies a logical block range whose data size exceeds the applicable recommended maximum may encounter delays in processing;

    and
b) if the NVMWZSV bit is cleared to '0', then:
    - the WZDSL field indicates the maximum data size limit for any Write Zeroes command that has the Deallocate bit set to '1';
    - the WZSL field indicates the maximum data size limit for any Write Zeroes command that has the Deallocate bit cleared to '0'; and

- the controller shall abort any Write Zeroes command that specifies a logical block range whose data size exceeds the applicable data size limit with a status code of Invalid Field in Command.

The WZSL field and the WZDSL field do not apply to Write Zeroes commands that have the Namespace Zeroes (NSZ) bit (refer to Figure 82) set to '1'.

The Namespace Zeroes (NSZ) bit is set to '1' to request that the controller clear all of the logical blocks to zero in the entire specified namespace by deallocating all logical blocks in that namespace. This functionality is only supported when the Deallocate bit is also set to '1' and the specified namespace supports clearing all bytes to 0h in the values read from a deallocated logical block and its metadata (excluding protection information) (e.g., as indicated by the DLFEAT field in Identify Namespace data structure being set to 001b (refer to Figure 114)).

If the NSZ bit is set to '1' in a Write Zeroes command and either:

- the Deallocate bit is cleared to '0'; or
- the specified namespace does not support clearing all bytes to 0h in the values read from a deallocated logical block and its metadata (excluding protection information),

then the controller shall abort that command with a status code of Invalid Field in Command.

Controller support for the NSZ bit is indicated by setting the NSZS bit to '1' in the ONCS field in the Identify Controller data structure. If the controller supports the NSZ bit and the NSZ bit is set to '1' in a Write Zeroes command, then the controller ignores the Starting LBA (SLBA) field (refer to Figure 81) and the Number of Logical Blocks (NLB) field (refer to Figure 82) in that command.

If the controller does not support the NSZ bit (i.e., the NSZS bit in the ONCS field in the Identify Controller data structure is cleared to '0'), then the controller is not required to check whether the NSZ bit is cleared to '0'; and may clear to zero the range of logical blocks specified by the SLBA field and NLB field instead of clearing all of the logical blocks to zero in the entire namespace.

The host is able to determine whether or not the controller cleared all of the logical blocks to zero in the entire specified namespace by checking the LBAs Cleared to Zero (LBACZ) bit of Dword 0 in the completion queue entry (CQE) for the Write Zeroes command (refer to Figure 87). The controller sets that bit to '1' if all of the logical blocks have been cleared to zero in the entire specified namespace (refer to section 3.3.8.1). The host should check this bit for any Write Zeroes command in which the NSZ bit is set to '1' to determine whether or not the controller cleared all of the logical blocks to zero in the entire specified namespace.

The fields used are Command Dword 2, Command Dword 3, Command Dword 10, Command Dword 11, Command Dword 12, Command Dword 13, Command Dword 14, and Command Dword 15 fields.

Figure 80: Write Zeroes – Command Dword 2 and Dword 3

Bits	Description
63:48	Reserved
47:00	**Logical Block Tags Upper (LBTU):** This field and Command Dword 14 specify the variable sized Logical Block Storage Tag (LBST) and Initial Logical Block Reference Tag (ILBRT) fields, which are defined in section 5.3.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller.

Figure 81: Write Zeroes – Command Dword 10 and Command Dword 11

Bits	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. If the NSZ bit (refer to Figure 82) is set to '1' and NSZS bit is set to '1' in the Optional NVMe Command Support (ONCS) field in the Identify Controller data structure, then this field should be cleared to 0h by the host and shall be ignored by the controller.

Figure 82: Write Zeroes – Command Dword 12

Bits	Description
31	**Limited Retry (LR):** If this bit is set to '1', then the controller should apply limited retry efforts. If this bit is cleared to '0', then the controller should apply all available error recovery means to write the data to the NVM.
30	**Force Unit Access (FUA):** If this bit is set to '1', then the controller shall write the data, and metadata, if any, to non-volatile medium before indicating command completion. There is no implied ordering with other commands. If this bit is cleared to '0', then this bit has no effect.
29:26	**Protection Information (PRINFO):** Specifies the protection information action and check field, as defined in Figure 11. The Protection Information Check (PRCHK) field shall be cleared to 000b.
25	**Deallocate (DEAC):** If this bit is set to '1', then the host is requesting that the controller deallocate the specified logical blocks. If this bit is cleared to '0', then the host is not requesting that the controller deallocate the specified logical blocks. If the NSZ bit is set to '1' and this bit is cleared to '0', then the controller shall abort the command with a status code of Invalid Field in Command.
24	**Storage Tag Check (STC):** This bit specifies the Storage Tag field shall be checked as part of end-to-end data protection processing as defined in Figure 12. This bit shall be cleared to '0'.
23	**Namespace Zeroes (NSZ):** If this bit is set to '1' and the Deallocate bit is set to '1', then the Write Zeroes command is requesting that the controller clear all logical blocks to zero in the entire namespace. If bit NSZS in the Optional NVM Command Support (ONCS) field in the Identify Controller data structure is cleared to '0', then this bit has no effect.
22:20	**Directive Type (DTYPE):** Specifies the Directive Type associated with the Directive Specific field (refer to the Directives section in the NVM Express Base Specification).
19:16	**Command Extension Type (CETYPE):** Specifies the Command Extension Type that applies to the command (refer to the Key Per I/O section in the NVM Express Base Specification).
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. If the NSZ bit is set to '1', then this field should be cleared to 0h by the host and shall be ignored by the controller.

The definition of Command Dword 13 is based on the CETYPE value. If the CETYPE value is cleared to 0h, then Command Dword 13 is defined in Figure 83. If the CETYPE value is non-zero, then Command Dword 13 is defined in Figure 84.

Figure 83: Write Zeroes – Command Dword 13 if CETYPE is cleared to 0h

Bits	Description
31:16	**Directive Specific (DSPEC):** Specifies the Directive Specific value associated with the Directive Type field (refer to the Directives section in the NVM Express Base Specification).
15:00	Reserved

Figure 84: Write Zeroes – Command Dword 13 if CETYPE is non-zero

Bits	Description
31:16	**Directive Specific (DSPEC):** Specifies the Directive Specific value associated with the Directive Type field (refer to the Directives section in the NVM Express Base Specification).
15:00	**Command Extension Value (CEV):** The definition of this field is dependent on the value of the CETYPE field. Refer to the Key Per I/O section in the NVM Express Base Specification.

Figure 85: Write Zeroes – Command Dword 14

Bits	Description
31:00	**Logical Block Tags Lower (LBTL):** This field and bits 47:00 of Command Dword 2 and Dword 3 specify the variable sized Logical Block Storage Tag (LBST) and Initial Logical Block Reference Tag (ILBRT) fields, which are defined in section 5.3.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller.

Figure 86: Write Zeroes – Command Dword 15

Bits	Description
31:16	**Logical Block Application Tag Mask (LBATM):** This field indicates the Application Tag Mask value. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.
15:00	**Logical Block Application Tag (LBAT):** This field indicates the Application Tag value. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.3.

3.3.8.1 Command Completion

Upon completion of the Write Zeroes command, the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command.

If the command does not complete successfully (i.e., completes with a status code other than Successful Completion), then the controller may or may not have completed the requested clearing of logical blocks to zero.

If the command has the Namespace Zeroes (NSZ) bit cleared to '0' and completes successfully, then the logical blocks specified by the command have been cleared to zero. If the command has the Namespace Zeroes (NSZ) bit set to '1' and completes successfully, then the LBAs Cleared to Zero bit in completion queue entry Dword 0 indicates whether the specified logical blocks have been cleared to zero as defined in Figure 87.

Figure 87: Write Zeroes – Completion Queue Entry Dword 0

Bits	Description
31:01	Reserved
00	**LBAs Cleared to Zero (LBACZ):** If the command has the Namespace Zeroes (NSZ) bit set to '1' and completes successfully, then this bit is defined as follows: **Value** / **Definition** 0b – The number of logical blocks specified by the Number of Logical Blocks (NLB) field have been cleared to zero. 1b – All logical blocks in the entire namespace have been cleared to zero.

Write Zeroes command specific status values (i.e., SCT field set to 1h) are shown in Figure 88.

Figure 88: Write Zeroes – Command Specific Status Values

Value	Definition
81h	**Invalid Protection Information:** The Protection Information (PRINFO) field (refer to Figure 82) settings specified in the command are invalid for the Protection Information with which the namespace was formatted (refer to the PI field in Figure 90 and the DPS field in Figure 114) or the ILBRT field is invalid (refer to section 5.3.3).
82h	**Attempted Write to Read Only Range:** The LBA range specified contains read-only blocks. The controller shall not return this status value if the read-only condition on the media is a result of a change in the write protection state of a namespace (refer to the Namespace Write Protection section in the NVM Express Base Specification).

4 Admin Commands for the NVM Command Set

4.1 Admin Command behavior for the NVM Command Set

The Admin commands are as defined in the NVM Express Base Specification. The NVM Command Set specific behavior for Admin commands is described in this section.

4.1.1 Asynchronous Event Request command

The Asynchronous Event Request command operates as defined in the NVM Express Base Specification. In addition to the Asynchronous Events defined in the NVM Express Base Specification, the NVM Command Set defines the Asynchronous Events defined in this section.

Figure 89: Asynchronous Event Information – Notice

Value	Definition
00h	**Attached Namespace Attribute Changed:** The Attached Namespace Attribute Changed asynchronous event operates as defined in the NVM Express Base Specification with the following modifications. A controller shall not send this event if: a) the value in the Namespace Utilization field (refer to Figure 114) has changed, as this is a frequent event that does not require action by the host; or b) capacity information (i.e., the NUSE field and the NVMCAP field) returned in the Identify Namespace data structure (refer to Figure 114) changed as a result of an ANA state change.
05h	**LBA Status Information Alert:** The criteria for generating an LBA Status Information Alert Notice event have been met (refer to section 5.2.1). Information about Potentially Unrecoverable LBAs is available in the LBA Status Information log page (refer to section 4.1.4.5). To clear this event, the host issues a Get Log Page command with Retain Asynchronous Event bit cleared to '0' for the LBA Status Information log page.
09h	**Allocated Namespace Attribute Changed:** The Allocated Namespace Attribute Changed asynchronous event operates as defined in the NVM Express Base Specification with the following modifications. A controller shall not send this event if: a) the value in the Namespace Utilization field (refer to Figure 114) has changed, as this is a frequent event that does not require action by the host; or b) capacity information (i.e., the NUSE field and the NVMCAP field) returned in the Identify Namespace data structure (refer to Figure 114) changed as a result of an ANA state change.

4.1.2 Format NVM command

The Format NVM command operates as defined in the NVM Express Base Specification. The Format Index indicates a valid LBA Format from the LBA Format field in the Identify Namespace data structure (refer to section 5.5). Other NVM Command Set specific fields are defined in Figure 90.

For the NVM Command Set, if the Format NVM command results in a change of the logical block size for the namespace, then the resulting namespace size (i.e., NSZE) (refer to Figure 114) and the namespace capacity (i.e., NCAP) (refer to Figure 114) may differ from the values indicated prior to the processing of the Format NVM command.

If the LBA Format Extension Enable (LBAFEE) field is not set to 1h in the Host Behavior Support feature (refer to the Host Behavior Support section in the NVM Express Base Specification), then the controller aborts a Format NVM command with a status code of Invalid Namespace or Format that specifies an LBA format (refer to section 5.3.1) of, or specifies an individual namespace formatted with:

a) 16b Guard Protection Information with the STS field set to a non-zero value;
b) 32b Guard Protection Information; or
c) 64b Guard Protection Information.

If:

- the LBA Format Extension Enable (LBAFEE) field is set to 1h in the Host Behavior Support feature;
- the Format NVM command specifies an LBA format with the STS field set to a non-zero value; and
- at least one namespace affected by that command is not formatted with that same LBA format,

then the controller shall abort the command with a status code of Invalid Namespace or Format.

If Flexible Data Placement (refer to the NVM Express Base Specification) is enabled in the Endurance Group associated with the specified namespace, then:

- If any Reclaim Unit Handle utilized by the specified namespace is shared by other namespaces and the specified Format Index does not match the Format Index of the other namespaces, then the command shall be aborted with a status code of Invalid Format. Refer to section 4.1.6.

Figure 90: Format NVM – Command Dword 10 – NVM Command Set Specific Fields

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>08</td><td>Protection Information Location (PIL): If this bit is set to ‘1’ and protection information is enabled (refer to section 5.3), then protection information is transferred as the first bytes of metadata. If this bit is cleared to ‘0’ and protection information is enabled, then protection information is transferred as the last bytes of metadata. This setting is reported in the End-to-end Data Protection Type Settings (DPS) field of the Identify Namespace data structure and is constrained by the End-to-end Data Protection Capabilities (DPC) field of the Identify Namespace data structure. For implementations compliant with version 1.0 or later of the NVM Command Set Specification, this bit shall be cleared to ‘0’.</td></tr>
<tr><td>07:05</td><td>Protection Information (PI): This field specifies whether end-to-end data protection is to be enabled and if enabled, the type of protection information to use. 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>Type 1 protection information is enabled</td></tr>
<tr><td>010b</td><td>Type 2 protection information is enabled</td></tr>
<tr><td>011b</td><td>Type 3 protection information is enabled</td></tr>
<tr><td>100b to 111b</td><td>Reserved</td></tr>
</table>
If end-to-end data protected is enabled, the host specifies the appropriate protection information in Copy commands, Read commands, Verify commands, Write commands, and Compare commands.</td></tr>
<tr><td>04</td><td>Metadata Settings (MSET): This bit is set to ‘1’ if the metadata is transferred as part of an extended data LBA. This bit 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 bit shall be ignored by the controller.</td></tr>
</table>

4.1.3 Get Features & Set Features commands

Figure 91 defines the Features support requirements for I/O Controllers supporting the NVM Command Set.

Figure 91: Feature Identifiers – NVM Command Set

Feature Identifier	Persistent Across Power Cycle and Reset[1]	Uses Memory Buffer for Attributes	Description	Scope
03h	Yes	Yes	LBA Range Type	Namespace
05h	No	No	Error Recovery	Namespace
0Ah	No	No	Write Atomicity Normal	Controller
15h	No	No	LBA Status Information Report Interval	Controller
1Ch	Yes	Yes	Performance Characteristics	NVM subsystem Namespace

Notes:
1. This column is only valid if the feature is not saveable (refer to the NVM Express Base Specification). 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.

Figure 92 defines the Set Features command specific status values that are specific to the NVM Command Set specific Feature Identifiers used during Command Completion.

Figure 92: Set Features – Command Specific Status Values

Value	Definition
14h	**Overlapping Range:** This error is indicated if the LBA Range Type data structure has overlapping ranges.

4.1.3.1 LBA Range Type (Feature Identifier 03h)

This feature indicates the type and attributes of LBA ranges that are part of the specified namespace. If multiple Set Features commands for this feature are processed, then only information from the most recent successful command is retained (i.e., subsequent commands replace information provided by previous commands).

A Set Features command with the Feature Identifier set to 03h and the NSID field set to FFFFFFFFh shall be aborted with a status of Invalid Field in Command.

The LBA Range Type feature uses Command Dword 11 and specifies the type and attribute information in the data structure indicated in Figure 95. The data structure is 4,096 bytes in size and shall be physically contiguous.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 94 are returned in Dword 0 of the completion queue entry and the LBA Range Type data structure specified in Figure 95 is returned in the data buffer for that command.

Figure 93: LBA Range Type – Command Dword 11

Bits	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 by the controller for the Get Features command for this Feature.

Figure 94: LBA Range Type – Completion Queue Entry Dword 0

Bits	Description
31:06	Reserved
05:00	**Number of LBA Ranges (NUM):** This field indicates the number of valid LBA ranges returned in the data buffer for the command (refer to Figure 95). This is a 0's based value.

Each entry in the LBA Range Type data structure is defined in Figure 95. 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 controller is not required to perform validation checks on any of the fields in this data structure. The LBA ranges should not overlap and may be listed in any order (e.g., ordering by LBA is not required). If the controller checks for LBA range overlap and the controller detects an LBA range overlap, then the controller shall abort the command with a status code of Overlapping Range.

For a Get Features command, the controller may clear to zero all unused entries in the LBA Range Type data structure. For a Set Features command, the controller shall ignore all unused entries in the LBA Range Type data structure.

If the size of the namespace or the LBA format of the namespace changes, then the specified LBA ranges may not represent the expected locations in the NVM. After such a change, the host should ensure the intended LBAs are specified.

The default value for this feature shall clear the Number of LBA Ranges field to 0h (i.e., one LBA Range is present) and initialize the LBA Range Type data structure to contain a single entry with the:

- Type field cleared to 0h;
- Attributes field set to 1h;
- Starting LBA field cleared to 0h;
- Number of Logical Blocks field set to indicate the number of LBAs in the namespace; and
- GUID field cleared to 0h, or set to a globally unique identifier.

Figure 95: LBA Range Type – Data Structure Entry

<table>
<tr><th>Bytes</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>Definition</th></tr>
<tr><td>0h</td><td>General Purpose</td></tr>
<tr><td>1h</td><td>Filesystem</td></tr>
<tr><td>2h</td><td>RAID</td></tr>
<tr><td>3h</td><td>Cache</td></tr>
<tr><td>4h</td><td>Page or swap file</td></tr>
<tr><td>5h to 7Fh</td><td>Reserved</td></tr>
<tr><td>80h to FFh</td><td>Vendor Specific</td></tr>
</table></td></tr>
<tr><td>01</td><td>Attributes (ATTRB): Specifies attributes of the LBA range. Each bit defines an attribute.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:2</td><td>Reserved</td></tr>
<tr><td>1</td><td>Hide LBA Range (HLBAR): If this bit is set to '1', then the LBA range should be hidden from the OS / EFI / BIOS. If this bit is cleared to '0', then the area should be visible to the OS / EFI / BIOS.</td></tr>
<tr><td>0</td><td>LBA Range Overwriteable (LBARO): If this bit is set to '1', then the LBA range may be overwritten. If this bit is cleared to '0', then the area should not be overwritten.</td></tr>
</table></td></tr>
<tr><td>15:02</td><td>Reserved</td></tr>
</table>

Figure 95: LBA Range Type – Data Structure Entry

Bytes	Description
23:16	**Starting LBA (SLBA):** This field specifies the 64-bit logical block address of the first logical block that is part of this LBA range.
31:24	**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 (e.g., the value 0h specifies one block).
47:32	**Unique Identifier (GUID):** This field contains a global unique identifier, for use by the host, that uniquely specifies the type of this LBA range. Well known Types may be defined and published on the NVM Express website.
63:48	Reserved

4.1.3.2 Error Recovery (Feature Identifier 05h)

This Feature controls the error recovery attributes for the specified namespace. The attributes are specified in Command Dword 11.

If a Get Features command is submitted for this Feature, the attributes described in Figure 96 are returned in Dword 0 of the completion queue entry for that command.

Figure 96: Error Recovery – Command Dword 11

Bits	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 specified namespace. If cleared to '0', then the Deallocated or Unwritten Logical Block error is disabled for the specified namespace. Host software shall only enable this error if the DAE bit in the NSFEAT field is set to '1' in the Identify Namespace data structure. The default value for this bit shall be '0'. Refer to section 3.3.3.2.1.
15:00	**Time Limited Error Recovery (TLER):** Indicates a limited retry timeout value in 100 millisecond units. This limit applies to I/O commands that support the Limited Retry bit and that are sent to the namespace for which this Feature has been set. 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.

4.1.3.3 Write Atomicity Normal (Feature Identifier 0Ah)

This Feature configures the controller operation of the AWUN and NAWUN parameters (refer to section 2.1.4.2). The attributes are specified in Command Dword 11.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 97 are returned in Dword 0 of the completion queue entry for that command.

Figure 97: Write Atomicity Normal – Command Dword 11

Bits	Description
31:01	Reserved
00	**Disable Normal (DN):** If this bit is 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 this bit is cleared to '0', then AWUN, NAWUN, AWUPF, and NAWUPF shall be honored by the controller.

4.1.3.4 Asynchronous Event Configuration (Feature Identifier 0Bh)

Figure 98: Asynchronous Event Configuration – NVM Command Set specific Bit Definitions

Bits	Description
13	**LBA Status Information Notices (LBASIN):** This bit determines whether an asynchronous event notification is sent to the host for an LBA Status Information Alert event (refer to Figure 89). If this bit is set to '1', then the LBA Status Information Alert event is sent to the host when this condition occurs. If this bit is cleared to '0', then the controller shall not send the LBA Status Information Alert event to the host.

4.1.3.5 LBA Status Information Attributes (Feature Identifier 15h)

The LBA Status Information Poll Interval (LSIPI) (refer to Figure 99) is the minimum interval that the host should wait between subsequent reads of the LBA Status Information log page with the Retain Asynchronous Event bit cleared to '0'. The LBA Status Information Poll Interval (LSIPI) is not changeable by the host.

The LBA Status Information Report Interval (LSIRI) (refer to Figure 99) is the minimum amount of time that a controller shall delay before sending an LBA Status Information Alert asynchronous event, if LBA Status Information Notices are enabled. The default value of the LSIRI is equal to LSIPI.

The host may read the LBA Status Information log page as part of LBA Status Information Alert asynchronous event processing or the host may use a polled method without enabling LBA Status Information Notices.

The controller reports the value of the LBA Status Information Attributes in Dword 0 of the completion queue entry when the host issues either a Set Features or Get Features command for this feature. The host configures the LBA Status Information Report Interval by issuing a Set Features command for this feature and specifying the value of the LBA Status Information Report Interval in Command Dword 11 (refer to Figure 99).

The host should not specify a value for the LBA Status Information Report Interval (LSIRI) which is less than the LBA Status Information Poll Interval (LSIPI) value reported by the controller. If the host specifies a value the controller does not support, the controller shall return the closest value supported by the controller in Dword 0 of the completion queue entry for the Set Features command. The accuracy of the interval measurement on the part of the controller is implementation specific.

The controller shall not send an LBA Status Information asynchronous event unless:

1. there are Tracked LBAs and:
    i. the LBA Status Information Report Interval condition has been exceeded and the LBA Status Generation Counter has been incremented since the last LBA Status Information Alert asynchronous event occurred; or
    ii. an implementation specific aggregate threshold, if any exists, of Tracked LBAs has been exceeded;

    or

2. a component (e.g., die or channel) failure has occurred that may result in the controller aborting commands with Unrecovered Read Error status.

When the host issues a Get Log Page command for Log Page Identifier 0Eh with the Retain Asynchronous Event bit cleared to '0', the LBA Status Information Alert asynchronous event is cleared, if one was outstanding, and the LBA Status Information Report Interval is restarted by the controller.

LBAs added to the Tracked LBA List or component failures that generate potential LBAs for an Untracked LBA list may be coalesced into a single LBA Status Information Alert asynchronous event notification.

Figure 99: LBA Status Information Attributes – Command Dword 11

Bits	Description
31:16	**LBA Status Information Poll Interval (LSIPI):** The minimum amount of time in 100 millisecond increments that the host should wait between subsequent reads of the LBA Status Information log page with the Retain Asynchronous Event bit cleared to '0'.
15:00	**LBA Status Information Report Interval (LSIRI):** If LBA Status Information Notices are enabled, the value in this field is the minimum amount of time in 100 millisecond increments that a controller shall delay before sending an LBA Status Information Alert asynchronous event.

4.1.3.6 Host Behavior Support (Feature Identifier 16h)

The Host Behavior Support feature operates as defined in the NVM Express Base Specification. In addition to the requirements in the NVM Express Base Specification, this specification provides NVM Command Set specific definitions.

Figure 100: Host Behavior Support – Data Structure

Bytes	Description
02	**LBA Format Extension Enable (LBAFEE):** This field allows the host to specify support for the extended LBA formats (refer to the ELBAS bit in the Identify Controller data structure in the NVM Express Base Specification). Refer to section 4.1.3.6.1 for further details. All values other than 0h and 1h are reserved.

4.1.3.6.1 LBA Format Extensions

The LBA Format Extension Enable (LBAFEE) field in the Host Behavior Support feature (refer to section 4.1.3.6) allows the host to enable support for extended protection information formats (refer to section 5.3.1) and for LBA Formats with Format Indexes greater than or equal to 16.

If the LBAFEE field is set to 1h and the ELBAS bit (refer to the Identify Controller data structure in the NVM Express Base Specification) is set to '1', then the controller:

a) shall report a maximum number that is less than or equal to 64 for:

   a. the total number of LBA formats supported (refer to section 5.5); and
   b. the number of namespace granularity descriptors (refer to Figure 123);

   and

b) is enabled to create namespaces with, format namespaces with, and perform I/O commands on namespaces with extended protection formats that are supported by the controller.

If the LBAFEE field is cleared to 0h, then the controller:

a) shall report a maximum number that is less than or equal to 16 for:

   a. the total number of LBA formats supported (refer to section 5.5); and
   b. the number of namespace granularity descriptors;

b) shall not create namespaces with, format namespaces with, or perform I/O commands on namespaces with extended protection information formats that are supported by the controller;
c) shall not format an individually specified namespace (refer to the Format NVM command section in the NVM Express Base Specification) that is formatted with an extended protection information format; and
d) shall not format or perform I/O commands on an individually specified namespace that is formatted with an LBA Format whose Format Index is greater than the reported total number of LBA formats supported (refer to section 5.5).

Commands that violate these restrictions shall be aborted with a status code of Invalid Namespace or Format.

4.1.3.7 Performance Characteristics (Feature Identifier 1Ch)

This Feature is used by the host to set and get Performance Characteristics information.

If a Get Features command for this feature specifying the SEL field set to 011b (i.e., Supported Capabilities) reports the NS Specific bit:

a) cleared to '0', then the NVM Subsystem Scope bit is set to '1' in the Feature Identifiers Supported and Effects log page (refer to NVM Express Base Specification); and
b) set to '1', then the Namespace Scope bit is set to '1' in the Feature Identifiers Supported and Effects log page.

Figure 101: Performance Characteristics – Command Dword 11

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:09</td><td>Reserved</td></tr>
<tr><td>08</td><td>Revert Vendor Specific Performance Attribute (RVSPA): If this bit is set to '1' in a Set Features command, then the saved attribute value of the Vendor Specific Performance Attribute specified by the Attribute Index field shall be deleted. If this bit is cleared to '0' in a Set Features command, then the saved attribute value of the Vendor Specific Performance Attribute specified by the Attribute Index field shall not be deleted.</td></tr>
<tr><td>07:00</td><td>Attribute Index (ATTRI): This field specifies the Performance Attribute to be transferred between the host and controller:
<table>
<tr><th>Value</th><th>Definition</th><th>Reference</th></tr>
<tr><td>00h</td><td>Standard Performance Attribute 00h</td><td>4.1.3.7.1</td></tr>
<tr><td>BFh to 01h</td><td>Reserved</td><td></td></tr>
<tr><td>C0h</td><td>Performance Attribute Identifier List</td><td>4.1.3.7.2</td></tr>
<tr><td>FFh to C1h</td><td>Vendor Specific Performance Attribute</td><td>4.1.3.7.3</td></tr>
</table>
</td></tr>
</table>

If a Set Features command is issued for this Feature and that command completes successfully, then the attribute specified by the Attribute Index field is transferred from the data buffer for that command.

If a Get Features command is issued for this Feature and that command completes successfully, then the attribute specified by the Attribute Index field and the Select field is returned in the data buffer for that command.

If a Get Features command or a Set Features command specifies an Attribute Index field with an unsupported value, then the controller shall abort that command with a status code of Invalid Field in Command.

If the Save and Select Feature Support (SSFS) bit is set to '1' in the Optional NVM Command Support (ONCS) field of the Identify Controller data structure and the value of the MSVSPA field is non-zero, then:

a) the capabilities for this Feature shall report changeable and saveable; and
b) a Set Features command specifying a Vendor Specific Performance shall specify a saved value. If a Set Features command is issued for this Feature and specifies:

- an Attribute Index field specifying a Vendor Specific Performance Attribute; and
- a Saved bit cleared to '0',

then the controller shall abort that command with a status code of Invalid Field in Command.

If a Set Features command specifies the Attribute Index of a Vendor Specific Performance Attribute that has a saved value and specifies the RVSPA bit set to '1', then:

- the saved attribute value of that Vendor Specific Performance Attribute is deleted;
- the contents of the data buffer are not used;
- the Save bit is ignored by the controller; and
- if that Vendor Specific Performance Attribute has not been set by an intervening Set Features command, then a subsequent Get Features command specifying that Vendor Specific Performance Attribute will return the default value for that Vendor Specific Performance Attribute.

If a Set Features command specifies the Attribute Index of a Vendor Specific Performance Attribute that does not have a saved value, and specifies the RVSPA bit set to '1', then:

- the contents of the data buffer are not used;
- the Save bit is ignored by the controller; and
- that command returns a status code of Successful Completion.

4.1.3.7.1 Standard Performance Attribute

The Random 4 KiB Average Read Latency field of the Standard Performance Attribute (refer to Figure 102) indicates the range corresponding to the value of the measured average latency. Average latency shall be measured according to the Latency Test section in the Solid State Storage (SSS) Performance Test Specification (refer to section 1.6). The Latency Test shall be performed using the PTS-C configuration parameters (WCE and AR=75). The measured latency is measured by the test loop for R/W% = 100/0 and Block Size = 4 KiB.

That specification describes reporting requirements for the test system used to perform the test. Reporting that information about the test system is outside the scope of this specification.

The Standard Performance Attribute is not able to be modified by a Set Features command. If a Set Features command is issued for the Feature and the Attribute Index field specifies the Standard Performance Attribute, then the controller shall abort that command with a status code of Invalid Field in Command.

Figure 102: Performance Characteristics – Standard Performance Attribute

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>03:00</td><td>Reserved</td></tr>
<tr><td>04</td><td>Random 4 KiB Average Read Latency (R4KARL): This field indicates the time to complete a 4 KiB random read. Each value indicates a range of latencies:
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00h</td><td>Not Reported</td></tr>
<tr><td>01h</td><td>Greater than or equal to 100 seconds</td></tr>
<tr><td>02h</td><td>Greater than or equal to 50 seconds and less than 100 seconds</td></tr>
<tr><td>03h</td><td>Greater than or equal to 10 seconds and less than 50 seconds</td></tr>
<tr><td>04h</td><td>Greater than or equal to 5 seconds and less than 10 seconds</td></tr>
<tr><td>05h</td><td>Greater than or equal to 1 second and less than 5 seconds</td></tr>
<tr><td>06h</td><td>Greater than or equal to 500 milliseconds and less than 1 second</td></tr>
<tr><td>07h</td><td>Greater than or equal to 100 milliseconds and less than 500 milliseconds</td></tr>
<tr><td>08h</td><td>Greater than or equal to 50 milliseconds and less than 100 milliseconds</td></tr>
<tr><td>09h</td><td>Greater than or equal to 10 milliseconds and less than 50 milliseconds</td></tr>
<tr><td>0Ah</td><td>Greater than or equal to 5 milliseconds and less than 10 milliseconds</td></tr>
<tr><td>0Bh</td><td>Greater than or equal to 1 millisecond and less than 5 milliseconds</td></tr>
<tr><td>0Ch</td><td>Greater than or equal to 500 microseconds and less than 1 millisecond</td></tr>
<tr><td>0Dh</td><td>Greater than or equal to 100 microseconds and less than 500 microseconds</td></tr>
<tr><td>0Eh</td><td>Greater than or equal to 50 microseconds and less than 100 microseconds</td></tr>
<tr><td>0Fh</td><td>Greater than or equal to 10 microseconds and less than 50 microseconds</td></tr>
<tr><td>10h</td><td>Greater than or equal to 5 microseconds and less than 10 microseconds</td></tr>
<tr><td>11h</td><td>Greater than or equal to 1 microsecond and less than 5 microseconds</td></tr>
<tr><td>12h</td><td>Greater than or equal to 500 nanoseconds and less than 1 microsecond</td></tr>
<tr><td>13h</td><td>Greater than or equal to 100 nanoseconds and less than 500 nanoseconds</td></tr>
<tr><td>14h</td><td>Greater than or equal to 50 nanoseconds and less than 100 nanoseconds</td></tr>
<tr><td>15h</td><td>Greater than or equal to 10 nanoseconds and less than 50 nanoseconds</td></tr>
<tr><td>16h</td><td>Greater than or equal to 5 nanoseconds and less than 10 nanoseconds</td></tr>
<tr><td>17h</td><td>Greater than or equal to 1 nanosecond and less than 5 nanoseconds</td></tr>
<tr><td>FFh to 18h</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>4095:05</td><td>Reserved</td></tr>
</table>

4.1.3.7.2 Performance Attribute Identifier List

The Performance Attribute Identifier List contains the Performance Attribute Identifiers of the Vendor Specific Performance Attributes, as described in Figure 103.

Figure 103: Performance Characteristics – Performance Attribute Identifier List

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>00</td><td>
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:3</td><td>Reserved</td></tr>
<tr><td>2:0</td><td>Attribute Type (ATTRTYP): Each Performance Attribute Identifier in this list is the value of the Performance Attribute Identifier field in the Vendor Specific Performance Attribute reported in response to a Get Features command specifying a Select field set to the value of this field.

The value of this field shall be equal to the value of the Select field of the Get Features command which specified this attribute.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>Current attribute</td></tr>
<tr><td>001b</td><td>Default attribute</td></tr>
<tr><td>010b</td><td>Saved attribute</td></tr>
<tr><td>All other values</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>01</td><td>Maximum Saveable Vendor Specific Performance Attributes (MSVSPA): This field indicates the maximum number of Vendor Specific Performance Attributes which are able to be saved.</td></tr>
<tr><td>02</td><td>Unused Saveable Vendor Specific Performance Attributes (USVSPA): This field indicates the number of saveable Vendor Specific Performance Attributes which have not been saved. This field shall be set to a value less than or equal to MSVSPA.</td></tr>
<tr><td>15:03</td><td>Reserved</td></tr>
<tr><th colspan="2">Performance Attribute Identifier List</th></tr>
<tr><td>31:16</td><td>Performance Attribute C1h Identifier (PAC1HI): Contains the Performance Attribute Identifier field of Vendor Specific Performance Attribute C1h.</td></tr>
<tr><td>47:32</td><td>Performance Attribute C2h Identifier (PAC2HI): Contains the Performance Attribute Identifier field of Vendor Specific Performance Attribute C2h.</td></tr>
<tr><td>…</td><td></td></tr>
<tr><td>1023:1008</td><td>Performance Attribute FFh Identifier (PACFFHI): Contains the Performance Attribute Identifier field of Vendor Specific Performance Attribute FFh.</td></tr>
<tr><td>4095:1024</td><td>Reserved</td></tr>
</table>

The value of the MSVSPA field indicates the number of Vendor Specific Performance Attributes which may be saved. The Attribute Indexes of Vendor Specific Performance Attributes which have been saved may be discontiguous in the range C1h to FFh.

If a Set Features command specifies a Vendor Specific Performance Attribute (i.e., an Attribute Index field in the range of C1h to FFh) and the value of the USVSPA field is cleared to 0h, then the controller shall abort that command with a status code of Invalid Field in Command.

The Performance Attribute Identifier List is reported separately for Current, Default, and Saved attributes. If the Save and Select Feature Support (SSFS) bit is set to ‘1’ in the Optional NVM Command Support (ONCS) field of the Identify Controller data structure, a Get Features command specifies an Attribute Index field set to C0h, and the Select field is set to:

- Current (i.e., 000b), then the controller reports a Performance Attribute Identifier List containing the Performance Attribute Identifier fields of the attributes reported in response to a Get Features command specifying a Select field set to Current and an Attribute Index set to a value in the range C1h to FFh;
- Default (i.e., 001b), then the controller reports a Performance Attribute Identifier List containing the Performance Attribute Identifier fields of the attributes reported in response to a Get Features

command specifying a Select field set to Default and an Attribute Index set to a value in the range C1h to FFh; or

- Saved (i.e., 010b), then the controller reports a Performance Attribute Identifier List containing the Performance Attribute Identifier fields of the attributes reported in response to a Get Features command specifying a Select field set to Saved and an Attribute Index set to a value in the range C1h to FFh.

The Performance Attribute Identifier List is not able to be modified by a Set Features command specifying an Attribute Index field set to C0h. If a Set Features command is issued for this feature and specifies an Attribute Index field set to C0h, then the controller shall abort that command with a status code of Invalid Field in Command.

4.1.3.7.3 Vendor Specific Performance Attribute

The Vendor Specific Performance Attribute is described in Figure 104:

Figure 104: Performance Characteristics – Vendor Specific Performance Attribute

Bytes	Description
15:00	**Performance Attribute Identifier (PAID):** This field contains an identifier describing the contents of the Vendor Specific field. Unused Vendor Specific Performance Attributes shall clear this field to 0h. It may be desirable for this field to be universally unique. In that case this field should be compatible with the 128-bit Universally Unique Identifier (UUID) specified in RFC 4122. Refer to the NVM Express Base Specification.
29:16	Reserved
31:30	**Attribute Length (ATTRL):** Indicates the number of valid bytes in the Vendor Specific field. The value shall be in the range 0h to FE0h.
4095:32	**Vendor Specific (VS)**

4.1.4 Get Log Page command

The Get Log Page command operates as defined in the NVM Express Base Specification. In addition to the requirements in the NVM Express Base Specification, mandatory, optional, and prohibited Log Page Identifiers are defined in Figure 105. If a Get Log Page command is processed that specifies a Log Page Identifier that is not supported, then the controller should abort the command with a status code of Invalid Field in Command.

In addition to the log pages described in the NVM Express Base Specification, the NVM Command Set defines the log pages described in this section. Log page scope is as defined in the NVM Express Base Specification, except as modified by this specification.

The rules for namespace identifier usage are specified in the NVM Express Base Specification.

Figure 105: Get Log Page – Log Page Identifiers

Log Page Identifier	CSI[1]	Scope	Log Page Name	Reference
01h	N	Controller	Error Information	4.1.4.1
02h	N	Controller or NVM subsystem	SMART / Health Information	4.1.4.2
06h	N	Controller	Device Self-test	4.1.4.3

Figure 105: Get Log Page – Log Page Identifiers

Log Page Identifier	CSI[1]	Scope	Log Page Name	Reference
0Eh	N	Controller	LBA Status Information	4.1.4.4

Notes:
1. If multiple I/O Command Sets are supported (refer to the NVM Express Base Specification), then the CSI field is used by the log page: Y = Yes, N = No. If Yes, then refer to the definition of the log page for details on usage.

4.1.4.1 Error Information (Log Page Identifier 01h)

The Error Information log page is as defined in the NVM Express Base Specification. Figure 106 describes the NVM Command Set specific definition of the LBA field.

Figure 106: Error Information Log Entry Data Structure – User Data

Bytes	Description
23:16	**Logical Block Address (LBA):** This field indicates the lowest-numbered LBA that experienced an error condition, if applicable.

4.1.4.2 SMART / Health Information (02h)

The SMART / Health Information log page is as defined in the NVM Express Base Specification. For the Data Units Read and Data Units Written fields, when the logical block size is a value other than 512 bytes, the controller shall convert the amount of data read to 512 byte units.

4.1.4.3 Device Self-test (Log Page Identifier 06h)

The Device Self-test log page is as defined in the NVM Express Base Specification. Figure 107 describes the NVM Command Set specific definition of the Failing LBA field.

Figure 107: Self-test Results Data Structure

Bytes	Description
23:16	**Failing LBA (FLBA):** 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'.

4.1.4.4 Persistent Event (Log Page Identifier 0Dh)

The Persistent Event log page is as defined in the NVM Express Base Specification. Figure 108 describes the NVM Command Set specific definition of the I/O Command Set specific fields within the Change Namespace Event Data Format (Event Type 06h) (refer to the NVM Express Base Specification).

Figure 108: Change Namespace Event Data Format (Event Type 06h)

Bytes	Description
32	**Formatted LBA Size (FLBAS):** For a create operation, contains the FLBAS value from the Host Software Specified Fields in the Namespace Management command (refer to Figure 125). For a delete operation that specifies a single namespace this field contains the value from the FLBAS field of the Identify Namespace data structure (refer to Figure 114) for the namespace being deleted. For a delete operation that specifies all namespaces this field is reserved.

Figure 108: Change Namespace Event Data Format (Event Type 06h)

Bytes	Description
33	**End-to-end Data Protection Type Settings (DPS):** For a create operation, contains the DPS value from the Host Software Specified Fields in the Namespace Management command (refer to Figure 125). For a delete operation that specifies a single namespace this field contains the value from the DPS field of the Identify Namespace data structure (refer to Figure 114) for the namespace being deleted. For a delete operation that specifies all namespaces this field is reserved.

4.1.4.5 LBA Status Information (Log Page Identifier 0Eh)

This log page is used to provide information about subsequent actions the host may take to discover which logical blocks, in namespaces that are attached to the controller, may no longer be recoverable when read. This log page contains zero or more LBA Status Log Namespace Elements (refer to Figure 110). If the controller is unaware of any potentially unrecoverable logical blocks in a given namespace attached to the controller, then this log page does not return an LBA Status Log Namespace Element for that namespace. This log page shall not return any LBA Status Log Namespace Elements for namespaces which are not attached to the controller.

Each LBA Status Log Namespace Element contains zero or more LBA Range Descriptors (refer to Figure 111). Each LBA Range Descriptor describes a range of LBAs that have been detected as being potentially unrecoverable and should be examined by the host using the mechanism specified in the Recommended Action Type field (refer to Figure 109) in that LBA Status Log Namespace Element in a subsequent Get LBA Status command.

The host may identify logical blocks that may no longer be recoverable through the subsequent issuing of one or more Get LBA Status commands (refer to section 4.2.1). Once identified, the host may then recover the user data from an alternative source and write that data to the original logical block address in the namespace. If the user data is written successfully, subsequent reads should not cause unrecoverable read errors (e.g., as a result of the write changing the physical location of the user data).

Upon receiving an LBA Status Information Alert asynchronous event, the host should send one or more Get Log Page commands for Log Page Identifier 0Eh with the Retain Asynchronous Event bit set to ‘1’ until the entire log page is read. To clear the event, the host sends a Get Log Page command for Log Page Identifier 0Eh with the Retain Asynchronous Event bit cleared to ‘0’. The host decides when to send Get LBA Status commands and when to recover the LBAs identified by the Get LBA Status commands, relative to when the host clears the event. Section 5.2.1.1 describes example host implementations. Clearing the event causes the LBA Status Information Report Interval to be restarted and allows the contents of the log page to be updated.

Figure 109: LBA Status Information Log Page

Bytes	Description
03:00	**LBA Status Log Page Length (LSLPLEN):** This field indicates the length in bytes of the LBA Status Information log page.
07:04	**Number of LBA Status Log Namespace Elements (NLSLNE):** This field indicates the number of LBA Status Log Namespace Elements (refer to Figure 110) contained in this log page. If this field is cleared to 0h and the Estimate of Unrecoverable Logical Blocks (ESTULB) field contains a non-zero value, the host should send Get LBA Status commands for the entire LBA range of each namespace attached to the controller. If both this field and the Estimate of Unrecoverable Logical Blocks (ESTULB) are cleared to 0h, the host should not send any Get LBA Status commands for any LBA ranges on any namespaces attached to the controller as there are no known potentially unrecoverable logical blocks in any namespace attached to the controller.

Figure 109: LBA Status Information Log Page

Bytes	Description
11:08	**Estimate of Unrecoverable Logical Blocks (ESTULB):** This field is an estimate of the sum of the total number of potentially unrecoverable logical blocks in all of the namespaces identified in the LBA Status Log Namespace Elements in this log page. A value of 0h in this field is valid. A value of FFFFFFFFh indicates no information regarding an estimate of the total number of potentially unrecoverable logical blocks is available.
13:12	Reserved
15:14	**LBA Status Generation Counter (LSGC):** Contains a value that is incremented each time the LBA Status Log contains one or more LBA Range Descriptors which specify any potentially unrecoverable logical blocks which were not included in any LBA Range Descriptors the last time the host read the LBA Status Information log. This field is persistent across power on. If the value of this field is FFFFh, then the field shall be cleared to 0h when incremented (i.e., rolls over to 0h).
n:16	**LBA Status Log Namespace Element List (LBASLNEL):** This field contains the list of LBA Status Log Namespace Elements that are present in the log page, if any. LBA Status Log Namespace Elements are of variable length (refer to Figure 110).

Figure 110: LBA Status Log Namespace Element

Bytes	Description
	Header
03:00	**Namespace Element Identifier (NEID):** This field indicates the Namespace Identifier (NSID) of the namespace that this LBA Status Log Namespace Element applies to.
07:04	**Number of LBA Range Descriptors (NLRD):** This field indicates the number of LBA Range Descriptors (refer to Figure 111) returned by the controller in this LBA Status Log Namespace Element. A value of FFFFFFFFh indicates that: a) no LBA Range Descriptor list is present; b) there is no information available regarding the location of known potentially unrecoverable blocks in the namespace; and c) the host should examine all LBAs in the namespace.
08	**Recommended Action Type (RATYPE):** This field indicates the value the host should set the Action Type (ATYPE) field to in Get LBA Status commands associated with LBA Range Descriptors contained in this LBA Status Log Namespace Element.
15:09	Reserved
	LBA Range Descriptor List
31:16	**LBA Range Descriptor 0:** This field contains the first LBA Range Descriptor in this LBA Status Log Namespace Element, if present.
47:32	**LBA Range Descriptor 1:** This field contains the second LBA Range Descriptor in this LBA Status Log Namespace Element, if present.
…	…
(NLRD*16+31): (NLRD*16+16)	**LBA Range Descriptor N LRD-1:** This field contains the last LBA Range Descriptor in this LBA Status Log Namespace Element, if present.

Figure 111: LBA Range Descriptor

Bytes	Description
07:00	**Range Starting LBA (RSLBA):** This field specifies the 64-bit address of the first logical block of this LBA Range.
11:08	**Range Number of Logical Blocks (RNLB):** This field contains the number of logical blocks in this LBA Range. The controller should return the largest possible value in this field. This is a 0's based value.
15:12	Reserved

For a given LBA Status Log Namespace Element, if the value in the Recommended Action Type (RATYPE) field is 10h, then the controller shall not report the same LBA Status Log Namespace Element once the host issues a Get Log Page command for Log Page Identifier 0Eh with the Retain Asynchronous Event bit cleared to '0' unless an additional component failure has occurred that may have created additional Untracked LBAs.

4.1.4.6 Flexible Data Placement (FDP) Statistics (Log Page Identifier 22h)

The Flexible Data Placement Statistics log page is defined in the NVM Express Base Specification. The I/O commands that are specifically used by the NVM Command Set for the Host Bytes with Metadata Written (HBMW) field and the Media Bytes with Metadata Written (MBMW) field are the User Data Out Commands, the Write Zeroes command, and the Write Uncorrectable command.

4.1.4.7 Flexible Data Placement (FDP) Events (Log Page Identifier 23h)

4.1.4.7.1 Controller Events

4.1.4.7.1.1 Media Reallocated (Event Type 0h)

The Flexible Data Placement Events log page is defined in the NVM Express Base Specification. Figure 112 describes the NVM Command Set specific definition of the Media Reallocated event.

Figure 112: Media Reallocated - Event Type Specific Data Structure

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>00</td><td>Specific Event Flags (SEF): This field indicates specific attributes of the event.
<table><tr><th>Bits</th><th>Description</th></tr><tr><td>07:01</td><td>Reserved</td></tr><tr><td>00</td><td>LBA Valid (LBAV): If this bit is set to '1', then the LBA field contains a valid value. If this bit is cleared to '0', then the LBA field does not contain a valid value.</td></tr></table></td></tr>
<tr><td>01</td><td>Reserved</td></tr>
<tr><td>03:02</td><td>Number of LBAs Moved (NLBAM): This field indicates the number of LBAs moved by the controller from a Reclaim Unit written by the host. If the PIV bit is set to '1', then the Placement Identifier field indicates the Reclaim Unit Handle used to initially write the LBAs. A value of 0h in this field indicates that no number is being reported. A value of FFFFh means that FFFFh or more LBAs were moved by the controller.</td></tr>
<tr><td>11:04</td><td>Logical Block Address (LBA): This field indicates one of the LBAs moved by the controller. If the PIV bit is set to '1', then the Placement Identifier field indicates the Reclaim Unit Handle used to initially write the LBAs.

If the LBAV bit is cleared to '0', then this field shall be cleared to 0h and the host should ignore this field.</td></tr>
<tr><td>15:12</td><td>Reserved</td></tr>
</table>

4.1.5 Identify Command

This specification implements the Identify Command and associated Identify data structures defined in the NVM Express Base Specification. Additionally, the NVM Command Set specifies the data structures defined in this section. The following table lists the Identify data structures that are added or modified by the NVM Command Set.

Each I/O Command Set is assigned a specific Command Set Identifier (CSI) value by the NVM Express Base Specification. The NVM Command Set is assigned a CSI value of 00h.

Figure 113: CNS Values

CNS Value	O/M[1]	Definition	NSID[2]	CNTID[3]	CSI[4]	Reference Section
Active Namespace Management						
00h	M	Identify Namespace data structure for the specified NSID or the namespace capabilities for the NVM Command Set. [6]	Y	N	N	4.1.5.1
01h	M	Identify Controller data structure for the controller processing the command. [6]	N	N	N	4.1.5.2
05h	M [5]	Identify I/O Command Set specific Namespace data structure for the specified NSID for the I/O Command Set specified in the CSI field. [6]	Y	N	Y	4.1.5.3
08h	M	Identify I/O Command Set specific Controller data structure for the controller processing the command. [6]	N	N	Y	4.1.5.4
09h	O	Identify Namespace data structure for the specified Format Index containing the namespace capabilities for the NVM Command Set. [6]	N	N	Y	4.1.5.5
0Ah	O	I/O Command Set specific Identify Namespace data structure for the specified Format Index for the I/O Command Set specified in the CSI field. [6]	N	N	Y	4.1.5.6
11h	O [5]	Identify Namespace data structure for the specified allocated NSID.	Y	N	N	4.1.5.7
16h	O	A Namespace Granularity List (refer to Figure 123) is returned to the host that contains up to sixteen Namespace Granularity Entries.	N	N	N	4.1.5.8
1Bh	O [5]	I/O Command Set specific Identify Namespace data structure for the specified allocated NSID.	Y	N	Y	4.1.5.9

Notes:
1. O/M definition: O = Optional, M = Mandatory.
2. The NSID field is used: Y = Yes, N = No.
3. The CDW10.CNTID field is used: Y = Yes, N = No.
4. The CDW11.CSI field is used: Y = Yes, N = No.
5. Mandatory for controllers that support the Namespace Management capability (refer to the NVM Express Base Specification).
6. Selection of a UUID may be supported. Refer to the Universally Unique Identifiers (UUIDs) for Vendor Specific Information section in the NVM Express Base Specification.

4.1.5.1 NVM Command Set Identify Namespace Data Structure (CNS 00h)

If the Namespace Identifier (NSID) field specifies an active NSID, then the NVM Command Set Identify Namespace data structure (refer to Figure 114) is returned to the host for that specified namespace. If that value in the NSID field is an inactive NSID, then the controller returns a zero filled data structure. If the specified namespace is not associated with an I/O Command Set that supports this data structure, then the controller shall abort the command with a status code of Invalid I/O Command Set.

The Reported column in Figure 114 specifies fields in the NVM Command Set Identify Namespace data structure that define namespace capabilities used by a host to format or create a namespace. If the NSID field is set to FFFFFFFFh, then the controller shall return an NVM Command Set Identify Namespace data structure that:

- for fields in Figure 114 that indicate “Yes” in the Reported column, contain a value that is the same for all namespaces using any of the LBA formats associated with the Number of LBA Formats field (refer to section 5.5); and
- for fields in Figure 114 that indicate “No” in the Reported column, contain a value cleared to 0h.

If the controller supports the Namespace Management capability (refer to the Namespace Management section in the NVM Express Base Specification) and the NSID field is set to FFFFFFFFh, then the controller shall return an NVM Command Set Identify Namespace data structure. If the controller does not support the Namespace Management capability and the NSID field is set to FFFFFFFFh, then the controller may abort the command with a status code of Invalid Namespace or Format.

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

Bytes	O/M[1]	Description	Reported[2]
07:00	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 LBA (*n* - 1). The number of logical blocks is based on the formatted logical block size. Refer to section 2.1.1 for details on the usage of this field.	No
15:08	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 logical block size. Spare LBAs are not reported as part of this field. Refer to section 2.1.1 for details on the usage of this field.	No
23:16	M	**Namespace Utilization (NUSE):** This field indicates the current number of logical blocks allocated in the namespace. This field is less than or equal to the value of the Namespace Capacity field. The number of logical blocks is based on the formatted logical block size. Refer to section 2.1.1 for details on the usage of this field.	No

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

<table>
<tr><th>Bytes</th><th>O/M[1]</th><th>Description</th><th>Reported[2]</th></tr>
<tr><td>24</td><td>M</td><td>Namespace Features (NSFEAT): This field defines features of the namespace.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>07</td><td>Optional Read Performance (OPTRPERF): If this bit is set to ‘1’ then the NPRG, BPRA, and NORS fields are defined for this namespace and should be used by the host for I/O optimization. If this bit is cleared to ‘0’, then the controller does not support the NPRG, NPRA, and NORS fields for this namespace.</td></tr>
<tr><td>06</td><td>Multiple Atomicity Mode (MAM): If this bit is set to ‘1’, then Multiple Atomicity Mode (refer to section 2.1.4.5) applies to write operations to this namespace. If this bit is cleared to ‘0’, then Single Atomicity Mode (refer to section 2.1.4.1) applies to write operations to this namespace.</td></tr>
<tr><td>05:04</td><td>Optional Write Performance (OPTPERF): Indicate support of alignment and granularity attributes of this namespace, as described in Figure 115.</td></tr>
<tr><td>03</td><td>UID Reuse (UIDREUSE): This bit is as defined in the UIDREUSE bit in the I/O Command Set Independent Identify Namespace data structure (refer to the I/O Command Set Independent Identify Namespace data structure section in the NVM Express Base Specification).</td></tr>
<tr><td>02</td><td>Deallocated Error (DAE): If this bit is set to ‘1’, then the controller supports the Deallocated or Unwritten Logical Block error for this namespace. If this bit is cleared to ‘0’, then the controller does not support the Deallocated or Unwritten Logical Block error for this namespace. Refer to section 3.3.3.2.1.</td></tr>
<tr><td>01</td><td>Namespace Supported Atomic Boundary & Power (NSABP): If this bit is set to ‘1’, then 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 this bit is 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 the NVM Express Base Specification. Refer to section 2.1.4.</td></tr>
<tr><td>00</td><td>Thin Provisioning (THINP): If this bit is set to ‘1’, then the namespace supports thin provisioning. If this bit is cleared to ‘0’, then thin provisioning is not supported Refer to section 2.1.1 for details on the usage of this bit.</td></tr>
</table></td><td>No</td></tr>
<tr><td>25</td><td>M</td><td>Number of LBA Formats (NLBAF): This field defines the number of supported LBA data size and metadata size combinations supported by the namespaces that share the same set of host-selectable attributes. LBA formats shall be packed sequentially starting at the LBA Format 0 Support (LBAF0) field. This is a 0’s based value.

Refer to section 5.5 for the structure of the LBA formats, the association to the NULBAF field, and the maximum values of this field.

The supported LBA formats are indicated in bytes 128 to 383 in this data structure. The LBA Format fields with a Format Index greater than the value defined by section 5.5 are invalid and not supported.

The metadata may be either transferred as part of the logical block or may be transferred as a separate contiguous buffer of data. The metadata shall not be split between the logical block and a separate metadata buffer. Refer to section 2.1.6.

It is recommended that software and controllers transition to an logical block size that is 4 KiB 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 5.2.3.</td><td>Yes</td></tr>
</table>

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

<table>
<tr><th>Bytes</th><th>O/M[1]</th><th>Description</th><th>Reported[2]</th></tr>
<tr><td>26</td><td>M</td><td>Formatted LBA Size (FLBAS): This field indicates the LBA data size and metadata size combination that the namespace has been formatted with (refer to section 4.1.2).
<table><tr><th>Bits</th><th>Description</th></tr><tr><td>7</td><td>Reserved</td></tr><tr><td>6:5</td><td>Format Index Upper (FIDXU): This field indicates the most-significant 2 bits of the Format Index that was used to format the namespace. If the total number of LBA formats supported (refer to section 5.5) is less than or equal to 16, then the host should ignore this field.</td></tr><tr><td>4</td><td>Metadata Transferred as Extended LBA (MTELBA): If this bit is set to ‘1’, then metadata is transferred at the end of the logical block, creating an extended logical block. If this bit is cleared to ‘0’, then indicates that all of the metadata for a command is transferred as a separate contiguous buffer of data. This bit is not applicable when there is no metadata.</td></tr><tr><td>3:0</td><td>Format Index Lower (FIDXL): This field indicates the least-significant 4 bits of the Format Index that was used to format the namespace.</td></tr></table></td><td>No</td></tr>
<tr><td>27</td><td>M</td><td>Metadata Capabilities (MC): This field indicates the capabilities for metadata.
<table><tr><th>Bits</th><th>Description</th></tr><tr><td>7:2</td><td>Reserved</td></tr><tr><td>1</td><td>Metadata Transferred as Separate Buffer Support (MTSBS): If this bit is set to ‘1’, then the namespace supports the metadata being transferred as part of a separate buffer that is specified in the Metadata Pointer. If this bit is cleared to ‘0’, then the namespace does not support the metadata being transferred as part of a separate buffer.</td></tr><tr><td>0</td><td>Metadata Transferred as Extended LBA Support (MTELBAS): If this bit is set to ‘1’, then the namespace supports the metadata being transferred as part of an extended data LBA. If this bit is cleared to ‘0’, then the namespace does not support the metadata being transferred as part of an extended data LBA.</td></tr></table></td><td>Yes</td></tr>
</table>

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

<table>
<tr><th>Bytes</th><th>O/M[1]</th><th>Description</th><th>Reported[2]</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 5.3.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:5</td><td>Reserved</td></tr>
<tr><td>4</td><td>Protection Information In Last Bytes (PIILB): If this bit is set to ‘1’, then the namespace supports protection information transferred as the last bytes of metadata. If this bit is cleared to ‘0’, then the namespace does not support protection information transferred as the last bytes of metadata.</td></tr>
<tr><td>3</td><td>Protection Information In First Bytes (PIIFB): If this bit is set to ‘1’, then the namespace supports protection information transferred as the first bytes of metadata. If this bit is cleared to ‘0’, then the namespace does not support protection information transferred as the first bytes of metadata. For implementations compliant with revision 1.0 or later of the NVM Command Set Specification, this bit shall be cleared to ‘0’.</td></tr>
<tr><td>2</td><td>Protection Information Type 3 Supported (PIT3S): If this bit is set to ‘1’, then the namespace supports Protection Information Type 3. If this bit is cleared to ‘0’, then the namespace does not support Protection Information Type 3.</td></tr>
<tr><td>1</td><td>Protection Information Type 2 Supported (PIT2S): If this bit is set to ‘1’, then the namespace supports Protection Information Type 2. If cleared to ‘0’ indicates that the namespace does not support Protection Information Type 2.</td></tr>
<tr><td>0</td><td>Protection Information Type 1 Supported (PIT1S): If this bit is set to ‘1’, then the namespace supports Protection Information Type 1. If this bit is cleared to ‘0’, then the namespace does not support Protection Information Type 1.</td></tr>
</table></td><td>Yes</td></tr>
<tr><td>29</td><td>M</td><td>End-to-end Data Protection Type Settings (DPS): This field indicates the protection information Type settings for the end-to-end data protection feature. Refer to section 5.3.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:4</td><td>Reserved</td></tr>
<tr><td>3</td><td>Protection Information Position (PIP): If this bit is set to ‘1’, then the protection information, if enabled, is transferred as the first bytes of metadata. If this bit is cleared to ‘0’, then the protection information, if enabled, is transferred as the last bytes of metadata. For implementations compliant with version 1.0 or later of the NVM Command Set Specification, this bit shall be cleared to ‘0’.</td></tr>
<tr><td>2:0</td><td>Protection Information Type (PIT): This field indicates 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>Type 1 protection information is enabled</td></tr>
<tr><td>010b</td><td>Type 2 protection information is enabled</td></tr>
<tr><td>011b</td><td>Type 3 protection information is enabled</td></tr>
<tr><td>100b to 111b</td><td>Reserved</td></tr>
</table></td></tr>
</table></td><td>No</td></tr>
<tr><td>30</td><td>O</td><td>Namespace Multi-path I/O and Namespace Sharing Capabilities (NMIC): This field is as defined in the I/O Command Set Independent Identify Namespace data structure (refer to the I/O Command Set Independent Identify Namespace data structure section in the NVM Express Base Specification).</td><td>Yes</td></tr>
</table>

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

<table>
<tr><th>Bytes</th><th>O/M[1]</th><th>Description</th><th>Reported[2]</th></tr>
<tr><td>31</td><td>O</td><td>Reservation Capabilities (RESCAP): This field is as defined in the I/O Command Set Independent Identify Namespace data structure (refer to the I/O Command Set Independent Identify Namespace data structure section in the NVM Express Base Specification).</td><td>No</td></tr>
<tr><td>32</td><td>O</td><td>Format Progress Indicator (FPI): This field is as defined in the I/O Command Set Independent Identify Namespace data structure (refer to the I/O Command Set Independent Identify Namespace data structure section in the NVM Express Base Specification).</td><td>No</td></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.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:5</td><td>Reserved</td></tr>
<tr><td>4</td><td>Guard Deallocation Status (GDS): If this bit is set to ‘1’, then 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 this bit is cleared to ‘0’, then each byte in the Guard field for the deallocated logical blocks that contain protection information is set to FFh.</td></tr>
<tr><td>3</td><td>Write Zeroes Deallocation Support (WZDS): If this bit is set to ‘1’, then the controller supports the Deallocate bit in the Write Zeroes command for this namespace. If this bit is cleared to ‘0’, then the controller does not support the Deallocate bit in the Write Zeroes command for this namespace. This bit shall be set to the same value for all namespaces in the NVM subsystem.</td></tr>
<tr><td>2:0</td><td>Deallocation Read Behavior (DRB): This field indicates the deallocated logical block read behavior. For a logical block that is deallocated, this field indicates the values read from that 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>The read behavior is not reported</td></tr>
<tr><td>001b</td><td>A deallocated logical block returns all bytes cleared to 0h</td></tr>
<tr><td>010b</td><td>A deallocated logical block returns all bytes set to FFh</td></tr>
<tr><td>011b to 111b</td><td>Reserved</td></tr>
</table></td></tr>
</table></td><td>No</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. If the NSABP bit is cleared to ‘0’, then this field is reserved.

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 (i.e., with the exception of the value 0, this field is a 0’s based value). Refer to section 2.1.4.</td><td>No</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. If the NSABP bit is cleared to ‘0’, then this field is reserved.

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 (i.e., with the exception of the value 0, this field is a 0’s based value). Refer to section 2.1.4.</td><td>No</td></tr>
</table>

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

Bytes	O/M[1]	Description	Reported[2]
39:38	O	**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. If the NSABP bit is cleared to '0', then this field is reserved. 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 (i.e., with the exception of the value 0, this field is a 0's based value). Refer to section 2.1.4.	No
41:40	O	**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 (i.e., with the exception of the value 0, this field is a 0's based value). Refer to section 2.1.4. Refer to section 5.2.2 for how this field is utilized.	No
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 2.1.4. Refer to section 5.2.2 for how this field is utilized.	No
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 (i.e., with the exception of the value 0, this field is a 0's based value). Refer to section 2.1.4.	No
47:46	O	**Namespace Optimal I/O Boundary (NOIOB):** This field indicates the optimal I/O boundary for this namespace. This field is specified in logical blocks. The host should construct read and write commands that do not cross the I/O boundary to achieve optimal performance. A value of 0h indicates that no optimal I/O boundary is reported. Refer to section 5.2.2 for how this field is utilized to improve performance and endurance.	No

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

Bytes	O/M[1]	Description	Reported[2]
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 the Namespace Management capability (refer to section 5.6) is 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 product of the logical block size and the Namespace Size reported. If the controller supports Asymmetric Namespace Access Reporting (refer to the CMIC field), and the relationship between the controller and the namespace is in the ANA Inaccessible state (refer to the ANA Inaccessible state section in the NVM Express Base Specification) or the ANA Persistent Loss state (refer to the ANA Persistent Loss state section in the NVM Express Base Specification), then this field shall be cleared to 0h.	No
65:64	O	**Namespace Preferred Write Granularity (NPWG):** This field indicates the smallest recommended write granularity in logical blocks for this namespace. This is a 0's based value. If this field is not supported as indicated by the OPTPERF field, then this field is reserved. The size indicated by this field should be less than or equal to the maximum number of logical blocks that are able to be transferred based on the value of the Maximum Data Transfer Size (MDTS) field defined in the Identify Controller data structure (refer to the NVM Express Base Specification). The MDTS field specified in units of minimum memory page size. The value of this field may change if the namespace is reformatted. The size should be a multiple of the Namespace Preferred Write Alignment (NPWA) field. Refer to section 5.2.2 for how this field is utilized to improve performance and endurance.	No
67:66	O	**Namespace Preferred Write Alignment (NPWA):** This field indicates the recommended write alignment in logical blocks for this namespace. This is a 0's based value. If this field is not supported as indicated by the OPTPERF field, then this field is reserved. The value of this field may change if the namespace is reformatted. Refer to section 5.2.2 for how this field is utilized to improve performance and endurance.	No
69:68	O	**Namespace Preferred Deallocate Granularity (NPDG):** This field indicates the recommended granularity in logical blocks for the Dataset Management command with the Attribute – Deallocate bit set to '1' in Dword 11. This is a 0's based value. If this field is not supported as indicated by the OPTPERF field, then this field is reserved. The value of this field may change if the namespace is reformatted. The size should be a multiple of the Namespace Preferred Deallocate Alignment (NPDA) field. Refer to section 5.2.2 for how this field is utilized to improve performance and endurance.	No
71:70	O	**Namespace Preferred Deallocate Alignment (NPDA):** This field indicates the recommended alignment in logical blocks for the Dataset Management command with the Attribute – Deallocate bit set to '1' in Dword 11. This is a 0's based value. If this field is not supported as indicated by the OPTPERF field, then this field is reserved. The value of this field may change if the namespace is reformatted. Refer to section 5.2.2 for how this field is utilized to improve performance and endurance.	No

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

<table>
<tr><th>Bytes</th><th>O/M[1]</th><th>Description</th><th>Reported[2]</th></tr>
<tr><td>73:72</td><td>O</td><td>Namespace Optimal Write Size (NOWS): This field indicates the size in logical blocks for optimal write performance for this namespace. This is a 0’s based value. If this field is not supported as indicated by the OPTPERF field, then this field is reserved.

If this namespace is associated with an NVM Set and:

a) this field is supported as defined by the OPTPERF field, then this field shall be set to the number of logical blocks corresponding to the Optimal Write Size field in the NVM Set Attributes Entry (refer to the Namespace Identification Descriptor in the NVM Express Base Specification) for that NVM Set; or
b) this field is not supported as defined by the OPTPERF field, then the host should use the Optimal Write Size field in the NVM Set Attributes Entry for that NVM Set for I/O optimization (refer to section 5.2.2).

The size indicated should be less than or equal to the maximum number of logical blocks that are able to be transferred based on the value of the Maximum Data Transfer Size (MDTS) field defined in the Identify Controller data structure (refer to the NVM Express Base Specification). The MDTS field is specified in units of minimum memory page size.The value of this field may change if the namespace is reformatted. The value of this field should be a multiple of the Namespace Preferred Write Granularity (NPWG) field.

Refer to section 5.2.2 for how this field is utilized to improve performance and endurance.</td><td>No</td></tr>
<tr><td>75:74</td><td>O</td><td>Maximum Single Source Range Length (MSSRL): This field indicates the maximum number of logical blocks that may be specified in the Number of Logical Blocks field in each valid Source Range Entries Descriptor of a Copy command (refer to section 3.3.2).

If the controller supports the Copy command, then this field shall be set to a non-zero value.</td><td>No</td></tr>
<tr><td>79:76</td><td>O</td><td>Maximum Copy Length (MCL): This field indicates the maximum number of logical blocks that may be specified in a Copy command (i.e., the sum of the number of logical blocks specified in all Source Range entries).

If the controller supports the Copy command, then this field shall be set to a non-zero value.</td><td>No</td></tr>
<tr><td>80</td><td>O</td><td>Maximum Source Range Count (MSRC): This field indicates the maximum number of Source Range entries that may be used to specify source data in a Copy command. This is a 0’s based value.</td><td>No</td></tr>
<tr><td>81</td><td>O</td><td>Key Per I/O Status (KPIOS): This field indicates namespace Key Per I/O capability status.

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:2</td><td>Reserved</td></tr>
<tr><td>1</td><td>Key Per I/O Supported in Namespace (KPIOSNS): If this bit is set to ‘1’, then the Key Per I/O capability is supported by the namespace. If this bit is cleared to ‘0’, then the Key Per I/O capability is not supported by the namespace.</td></tr>
<tr><td>0</td><td>Key Per I/O Enabled in Namespace (KPIOENS): If this bit is set to ‘1’, then the Key Per I/O capability is enabled on the namespace. The mechanism to enable the Key Per I/O capability on the namespace is outside the scope of this specification (refer to section 5.4). If this bit is cleared to ‘0’, then the Key Per I/O capability is disabled on the namespace.

If the KPIOSNS bit is cleared to ‘0’, then this bit shall be cleared to ‘0’.</td></tr>
</table></td><td>No</td></tr>
</table>

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

Bytes	O/M[1]	Description	Reported[2]
82	M	**Number of Unique Attribute LBA Formats (NULBAF):** This field defines the number of supported user data size and metadata size combinations supported by the namespace that may not share the same host-selectable attributes. These LBA formats shall be allocated in order (starting at the first index after the LBA formats defined by the NLBAF field) and packed sequentially (refer to section 5.5). Refer to section 5.5 for the structure of the LBA formats, the association to the NLBAF field, and the maximum value of this field.	Yes
83		Reserved	
87:84	O	**Key Per I/O Data Access Alignment and Granularity (KPIODAAG):** This field indicates the alignment and granularity in logical blocks that is required for commands that support a KPIOTAG value in the CETYPE field (refer to the Key Per I/O section in the NVM Express Base Specification). This is a 0's based value. Refer to section 5.4 on the behavior of commands not meeting the alignment or granularity defined by this field. The value of this field may change if the namespace is reformatted. If the KPIOSNS bit is cleared to '0' in the I/O Command Set Independent Identify Namespace data structure (refer to the NVM Express Base Specification), then this field is reserved.	No
91:88		Reserved	
95:92	O	**ANA Group Identifier (ANAGRPID):** This field is as defined in the I/O Command Set Independent Identify Namespace data structure (refer to the I/O Command Set Independent Identify Namespace data structure section in the NVM Express Base Specification).	No
98:96		Reserved	
99	O	**Namespace Attributes (NSATTR):** This field is as defined in the I/O Command Set Independent Identify Namespace data structure (refer to the I/O Command Set Independent Identify Namespace data structure section in the NVM Express Base Specification).	No
101:100	O	**NVM Set Identifier (NVMSETID):** This field is as defined in the I/O Command Set Independent Identify Namespace data structure (refer to the I/O Command Set Independent Identify Namespace data structure section in the NVM Express Base Specification).	No
103:102	O	**Endurance Group Identifier (ENDGID):** This field is as defined in the I/O Command Set Independent Identify Namespace data structure (refer to the I/O Command Set Independent Identify Namespace data structure section in the NVM Express Base Specification).	No

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

Bytes	O/M[1]	Description	Reported[2]
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 Level 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 identifier assigned by the corresponding organization. Bytes 111:104 contain the vendor specific extension identifier assigned by the corresponding organization. Refer to the IEEE EUI-64 guidelines for more information. This field is big endian (refer to the Namespace Globally Unique Identifier section in the NVM Express Base Specification). The controller shall specify a globally unique namespace identifier in this field, the EUI64 field, or a Namespace UUID in the Namespace Identification Descriptor (refer to the Namespace Identification Descriptor figure in the NVM Express Base Specification) when the namespace is created. If the controller is not able to provide a globally unique identifier in this field, then this field shall be cleared to 0h. Refer to the Unique Identifier section in the NVM Express Base Specification.	No
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 Level 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. Refer to the IEEE EUI-64 guidelines for more information. This field is big endian (refer to the IEEE Extended Unique Identifier section in the NVM Express Base Specification). The controller shall specify a globally unique namespace identifier in this field, the NGUID field, or a Namespace UUID in the Namespace Identification Descriptor (refer to the Namespace Identification Descriptor figure in the NVM Express Base Specification) when the namespace is created. If the controller is not able to provide a globally unique 64-bit identifier in this field, then this field shall be cleared to 0h. Refer to the Unique Identifier section in the NVM Express Base Specification.	No
LBA Formats (refer to section 5.5)			
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 116. Additional information may be provided in the ELBAF0 field (refer to Figure 118).	Yes
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 116. Additional information may be provided in the ELBAF1 field (refer to Figure 118).	Yes
…			
383:380	O	**LBA Format 63 Support (LBAF63):** This field indicates the LBA format 63 that is supported by the controller. The LBA format field is defined in Figure 116. Additional information may be provided in the ELBAF63 field (refer to Figure 118).	Yes
4095:384	O	**Vendor Specific (VS)**	No

Notes:
1. O/M definition: O = Optional, M = Mandatory.
2. Identifies fields that report information for the Identify command when querying the capabilities of LBA formats.

Figure 115: Namespace Alignment and Granularity Attributes

Field Supported	Optimal Write Performance Value			
	00b	01b	10b	11b
NPWG	No	Yes	Yes	Yes
NPWA	No	Yes	Yes	Yes
NPDG	No	Yes	No	Yes
NPDA	No	Yes	Yes	Yes
NPDGL	No	No	Yes	Yes
NPDAL	No	No	Yes	Yes
NOWS	No	Yes	Yes	Yes
The use of these fields by the host for I/O optimization is described in section 5.2.2.				

The LBA format data structure is described in Figure 116.

Figure 116: LBA Format Data Structure, NVM Command Set Specific

Bits	Description
31:26	Reserved
25:24	**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 4 KiB read workload. The meanings of the values indicated are included in the following table. **Value** / **Definition** 00b – Best performance 01b – Better performance 10b – Good performance 11b – Degraded performance
23:16	**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 non-zero value less than 9 (i.e., 512 bytes) is not supported. If the value reported is 0h, then the LBA format is not currently available (refer to section 5.5).
15:00	**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 0h. 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 (refer to the DPS field in the Identify Namespace data structure).

4.1.5.2 I/O Command Set specific fields within Identify Controller data structure (CNS 01h)

The following table describes the NVM Command Set specific fields within the Identify Controller data structure described in the NVM Express Base Specification.

Figure 117: Identify – Identify Controller data structure, NVM Command Set Specific Fields

Bytes	O/M[1]	Description
…		

Figure 117: Identify – Identify Controller data structure, NVM Command Set Specific Fields

Bytes	O/M[1]	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 2.1.4. If a write command is submitted that has a 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 that has a size greater than the AWUN value, then there is no guarantee of command atomicity, but atomicity is guaranteed for portions of the command if the command is processed in Multiple Atomicity Mode (refer to section 2.1.4.5). AWUN does not have any applicability to write errors caused by power failure (refer to Atomic Write Unit Power Fail). For any write command other than the Copy command, a value of FFFFh indicates the command is always atomic as this is the largest command size. For a Copy command, a value of FFFFh indicates that the atomicity of the write portion of the Copy command is 10000h logical blocks. It is recommended that implementations support a minimum of 128 KiB (appropriately scaled based on logical block 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 2.1.4. 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 that has a 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, but atomicity is guaranteed for portions of the command if the command is processed in Multiple Atomicity Mode (refer to section 2.1.4.5). 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 that has a size greater than the AWUPF value, then there is no guarantee of data returned on subsequent reads of the associated logical blocks.
…		

Figure 117: Identify – Identify Controller data structure, NVM Command Set Specific Fields

Bytes	O/M[1]	Description
533:532	O	**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 the Atomic Compare & Write Unit size for that namespace is reported in the NACWU field in the Identify Namespace data structure. Refer to section 2.1.4. 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 abort the command with a status code of Atomic Write Unit Exceeded. If Compare and Write is not a supported fused command, then this field shall be 0h.
…		
Notes: 1. O/M definition: O = Optional, M = Mandatory		

4.1.5.3 I/O Command Set Specific Identify Namespace Data Structure (CNS 05h)

Figure 118 defines the I/O Command Set specific Identify Namespace data structure for the NVM Command Set.

The Reported column in Figure 118 specifies fields in the I/O Command Set specific Identify Namespace data structure for the NVM Command Set that define namespace capabilities used by a host to format or create a namespace. If the NSID field is set to FFFFFFFFh, then the controller shall return an I/O Command Set specific Identify Namespace data structure for the NVM Command Set that:

- for fields in Figure 118 that indicate "Yes" in the Reported column, contain a value that is the same for all namespaces using any of the LBA formats associated with the Number of LBA Formats field (refer to section 5.5); and
- for fields in Figure 118 that indicate "No" in the Reported column, contain a value cleared to 0h.

If the controller supports the Namespace Management capability (refer to the Namespace Management section in the NVM Express Base Specification) and the NSID field is set to FFFFFFFFh, then the controller shall return an I/O Command Set specific Identify Namespace data structure for the NVM Command Set. If the controller does not support the Namespace Management capability and the NSID field is set to FFFFFFFFh, then the controller may abort the command with a status code of Invalid Namespace or Format.

Figure 118: NVM Command Set I/O Command Set Specific Identify Namespace Data Structure (CSI 00h)

Bytes	O/M[1]	Description	Reported[2]
7:0	O	**Logical Block Storage Tag Mask (LBSTM):** Indicates the mask for the Storage Tag field for the protection information (refer to section 5.3). The size of the mask contained in this field is defined by the STS field (refer to Figure 119). If the size of the mask contained in this field is less than 64 bits, the mask is contained in the least-significant bits of this field. The host should ignore bits in this field that are not part of the mask. If end-to-end protection is not enabled in the namespace, then this field should be ignored by the host. If: a) the Qualified Protection Information Format Support bit is set to '1'; and b) the Storage Tag Masking Level Attribute field is set to a value of 010b (i.e., Masking Not Supported); or a) end-to-end protection is enabled; b) 16b Guard Protection Information format is used; and c) the 16BPISTM bit is set to '1' in the PIC field, then each bit in the mask in this field shall be set to '1'. If the Qualified Protection Information Format Support bit is set to '1', then the Storage Tag Masking Level Attribute field imposes constraints on how the bits in the mask contained in this field are allowed to be configured.	No

Figure 118: NVM Command Set I/O Command Set Specific Identify Namespace Data Structure (CSI 00h)

<table>
<tr><th>Bytes</th><th>O/M[1]</th><th>Description</th><th>Reported[2]</th></tr>
<tr><td>8</td><td>O</td><td>Protection Information Capabilities (PIC): This field indicates the capabilities for the protection information formats.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:4</td><td>Reserved</td></tr>
<tr><td>3</td><td>Qualified Protection Information Format Support (QPIFS): If this bit is set to ‘1’, then the namespace supports the Qualified Protection Information Format field (refer to Figure 119) and the Storage Tag Masking Level Attribute field. If this bit is cleared to ‘0’, then the namespace does not support the Qualified Protection Information Format field (refer to Figure 119) and does not support the Storage Tag Masking Level Attribute field.</td></tr>
<tr><td>2</td><td>Storage Tag Check Read Support (STCRS): If this bit is set to ‘1’, then the controller supports the Storage Tag Check Read (STCR) bit in the Copy command (refer to Figure 34). If this bit is cleared to ‘0’, the controller does not support the Storage Tag Check Read bit in the Copy command. If the 16b Guard Protection Information Storage Tag Support (PISTS16B) bit is set to ‘1’, then this bit shall be set to ‘1’.</td></tr>
<tr><td>1</td><td>16b Guard Protection Information Storage Tag Mask (PISTM16B): If this bit is set to ‘1’, then the LBSTM field shall have all bits set to ‘1’ for the 16b Guard Protection Information. If this bit is cleared to ‘0’, then the Logical Block Storage Tag Mask field is allowed to have any bits set to ‘1’ for the 16b Guard Protection Information.

If the PISTS16B bit is cleared to ‘0’, then the PISTM16B bit should be ignored by the host.

If the Qualified Protection Information Format Support bit is set to ‘1’, the PIF field is set to 11b (i.e., Qualified Type), and the Storage Tag Masking Level Attribute is set to 010b (i.e., Masking Not Supported), then the PISTM16B bit shall be set to ‘1’.

The previous abbreviation for this bit was 16BPISTM.</td></tr>
<tr><td>0</td><td>16b Guard Protection Information Storage Tag Support (PISTS16B): If this bit is set to ‘1’, then the end-to-end protection 16b Guard Protection Information format (refer to section 5.3.1.1) supports a non-zero value in the STS field. If this bit is cleared to ‘0’, then the end-to-end protection 16b Guard Protection Information format support requires that the STS field be cleared to 0h (i.e., the Storage Tag field is not supported).

If the 32b Guard Protection Information or 64b Guard Protection Information is supported in any LBA format (refer to Figure 119), then this bit shall be set to ‘1’.

The previous abbreviation for this bit was 16BPISTS.</td></tr>
</table></td><td>Yes</td></tr>
</table>

Figure 118: NVM Command Set I/O Command Set Specific Identify Namespace Data Structure (CSI 00h)

<table>
<tr><th>Bytes</th><th>O/M[1]</th><th>Description</th><th>Reported[2]</th></tr>
<tr><td>9</td><td>O</td><td>Protection Information Format Attribute (PIFA): This field indicates attributes of the Protection Information Format supported by the namespace.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:3</td><td>Reserved</td></tr>
<tr><td>2:0</td><td>Storage Tag Masking Level Attribute (STMLA): This field indicates the type of storage tag masking the namespace supports:
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>Bit Granularity Masking: Unless otherwise specified, the bits in the Logical Block Storage Tag Mask fields (refer to Figure 118 and Figure 125) may be any combination of ‘1’s and ‘0’s.</td></tr>
<tr><td>001b</td><td>Byte Granularity Masking: The value of all bits within any individual byte in the Logical Block Storage Tag Mask fields (refer to Figure 118 and Figure 125) shall be the same, but that value may differ from one byte to another, and the value of all bits within any partial high-order byte that may exist in the Logical Block Storage Tag Mask fields shall be the same.</td></tr>
<tr><td>010b</td><td>Masking Not Supported: Each bit in the Logical Block Storage Tag Mask field (refer to Figure 125) is required to be set to ‘1’ when creating a namespace using the Namespace Management command (refer to section 4.1.6).</td></tr>
<tr><td>011b to 111b</td><td>Reserved</td></tr>
</table>
If the Qualified Protection Information Format Support bit (refer to Figure 118) is cleared to ‘0’ or the PIF field is set to a value other than 11b (i.e., other than Qualified Type), then this field shall be cleared to 000b.</td></tr>
</table></td><td>Yes</td></tr>
<tr><td>11:10</td><td></td><td>Reserved</td><td></td></tr>
<tr><td colspan="4">Extended LBA Format (refer to section 5.5)</td></tr>
<tr><td>15:12</td><td>O</td><td>Extended LBA Format 0 Support (ELBAF0): This field indicates additional LBA Format 0 information related to the LBA Format 0 Support (LBAF0) field in the Identify Namespace data structure. The Extended LBA format field is defined in Figure 119.</td><td>Yes</td></tr>
<tr><td>19:16</td><td>O</td><td>Extended LBA Format 1 Support (ELBAF1): This field indicates additional LBA Format 1 information related to the LBA Format 1 Support (LBAF1) field in the Identify Namespace data structure. The Extended LBA format field is defined in Figure 119.</td><td>Yes</td></tr>
<tr><td>…</td><td></td><td></td><td></td></tr>
<tr><td>267:264</td><td>O</td><td>Extended LBA Format 63 Support (ELBAF63): This field indicates additional LBA Format 63 information related to the LBA Format 63 Support (LBAF63) field in the Identify Namespace data structure. The Extended LBA format field is defined in Figure 119.</td><td>Yes</td></tr>
</table>

Figure 118: NVM Command Set I/O Command Set Specific Identify Namespace Data Structure (CSI 00h)

Bytes	O/M[1]	Description	Reported[2]
271:268	O	**Namespace Preferred Deallocate Granularity Large (NPDGL):** This field indicates the recommended granularity in logical blocks for the Dataset Management command with the Attribute – Deallocate bit set to '1' in Command Dword 11. If this field is not supported as defined by the OPTPERF field (refer to Figure 114), then this field is reserved. If this field is cleared to 0h, then this field does not indicate a recommended granularity. The value of this field may change if the namespace is reformatted. If the Namespace Preferred Deallocate Alignment Large (NPDAL) field is cleared to 0h, then the value of the NPDGL field should be a multiple of the Namespace Preferred Deallocate Alignment (NPDA) field (refer to Figure 114). If the Namespace Preferred Deallocate Alignment Large (NPDAL) field is supported as defined by the OPTPERF field (refer to Figure 114) and is set to a non-zero value, then the value of the NPDGL field should be a multiple of the NPDAL field. Refer to section 5.2.2 for how this field is utilized to improve performance and endurance.	No
275:272	O	**Namespace Preferred Read Granularity (NPRG):** This field is the smallest recommended read granularity in logical blocks for this namespace. This is a 0's based value. If this field is not supported as indicated by the OPTRPERF field, then this field is reserved. The size indicated by this field should be less than or equal to the size indicated by the Maximum Data Transfer Size (MDTS) field (refer to the NVM Express Base Specification) which is specified in units of minimum memory page size. The value of this field may change if the namespace is reformatted. The size should be a multiple of the Namespace Preferred Read Alignment (NPRA). Refer to section 5.2.2 for how this field is utilized to improve performance.	No
279:276	O	**Namespace Preferred Read Alignment (NPRA):** This field indicates the recommended read alignment in logical blocks for this namespace (refer to section 5.2.2.3). This is a 0's based value. If this field is not supported as indicated by the OPTRPERF field, then this field is reserved. The value of this field may change if the namespace is reformatted. Refer to section 5.2.2 for how this field is utilized to improve performance.	No
283:280	O	**Namespace Optimal Read Size (NORS):** This field indicates the size in logical blocks for optimal read performance for this namespace. This is a 0's based value. If this field is not supported as indicated by the OPTRPERF field, then this field is reserved. 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. The value of this field may change if the namespace is reformatted. The value of this field should be a multiple of Namespace Preferred Read Granularity (NPRG). Refer to section 5.2.2 for how this field is utilized to improve performance and endurance.	No

Figure 118: NVM Command Set I/O Command Set Specific Identify Namespace Data Structure (CSI 00h)

Bytes	O/M[1]	Description	Reported[2]
287:284	O	**Namespace Preferred Deallocate Alignment Large (NPDAL):** This field indicates the recommended alignment in logical blocks for the Dataset Management command with the Attribute – Deallocate bit set to '1' in Dword 11. If this field is not supported as indicated by the OPTPERF field, then this field is reserved. The value of this field may change if the namespace is reformatted. Refer to section 5.2.2 for how this field is utilized to improve performance and endurance.	No
291:288	O	**LBA Format Placement Shard Size (LBAPSS):** This field indicates the optimal number of LBAs to be written to a Reclaim Group and then written to the other Reclaim Groups in order to maximize I/O write performance. A value of 0h means that no Placement Shard Size value is reported.	No
295:292	O	**Tracked LBA Allocation Granularity (TLBAAG):** This field indicates the alignment and granularity, in logical blocks, for the reporting of allocated LBAs for the namespace by the Get LBA Status command (refer to section 4.2.1). If this field is cleared to the value of 0h, then the alignment and granularity are not reported. It is recommended that value of this field multiplied by the logical block data size for this namespace be greater than or equal to 4 KiB.	Yes
4095:296	O	Reserved	

Notes:
1. O/M definition: O = Optional, M = Mandatory.
2. Identifies fields that report information for the Identify command when querying the capabilities of LBA formats.

The Extended LBA format data structure is described in Figure 119.

Figure 119: Extended LBA Format Data Structure, NVM Command Set Specific

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:13</td><td>Reserved</td></tr>
<tr><td>12:9</td><td>Qualified Protection Information Format (QPIF):

If:

a) the Protection Information Format (PIF) field is set to a value of 11b (i.e., Qualified Type); and
b) end-to-end protection information is enabled on a namespace formatted with this LBA format,

then

a) this field indicates the protection information format (refer to section 5.3.1); and
b) that protection information format is qualified by the Storage Tag Mask constraints, if any, indicated by the Storage Tag Masking Level Attribute field (refer to Figure 118).

If the PIF field is set to a value other than 11b (i.e., Qualified Type) then this field is ignored.

<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0h</td><td>16b Guard Protection Information</td></tr>
<tr><td>1h</td><td>32b Guard Protection Information</td></tr>
<tr><td>2h</td><td>64b Guard Protection Information</td></tr>
<tr><td>3h to Fh</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>

Figure 119: Extended LBA Format Data Structure, NVM Command Set Specific

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>8:7</td><td>Protection Information Format (PIF): This field indicates the protection information format (refer to section 5.3.1) when end-to-end protection information is enabled on a namespace formatted with this LBA format.

If:

• end-to-end protection information is not supported by this LBA format; or
• end-to-end protection is disabled on a namespace formatted with this LBA format,

then this field is undefined and should be ignored by the host.

<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>16b Guard Protection Information</td></tr>
<tr><td>01b</td><td>32b Guard Protection Information</td></tr>
<tr><td>10b</td><td>64b Guard Protection Information</td></tr>
<tr><td>11b</td><td>Qualified Type (QTYPE): If the Qualified Protection Information Format Support bit is set to ‘1’, then the protection information format is as defined in the Qualified Protection Information Format (QPIF) field. If the Qualified Protection Information Format Support bit is cleared to ‘0’, then this value shall not be used.</td></tr>
</table>
</td></tr>
<tr><td>6:0</td><td>Storage Tag Size (STS): Identifies the number of most significant bits of the protection information Storage and Reference Space field that define the Storage Tag field (refer to section 5.3.1.4).

This field does limit the minimum and maximum values allowed per protection information formats (refer to section 5.3.1):

<table>
<tr><th>Protection Information Format</th><th>Minimum Value</th><th>Maximum Value</th></tr>
<tr><td>16b Guard Protection Information</td><td>0</td><td>32</td></tr>
<tr><td>32b Guard Protection Information</td><td>16</td><td>64</td></tr>
<tr><td>64b Guard Protection Information</td><td>0</td><td>48</td></tr>
</table>
If this field is cleared to 0h, then no bits of the Storage and Reference Space field are applied to the Storage Tag field and therefore the Storage Tag field is not defined.

For the 16b Guard Protection, if this field is set to 32, then no bits of the Storage and Reference Space field are applied to the Reference Tag field and therefore the Reference Tag field is not defined.

For the 64b Guard Protection, if this field is set to 48, then no bits of the Storage and Reference Space field are applied to the Reference Tag field and therefore the Reference Tag field is not defined.</td></tr>
</table>

4.1.5.4 I/O Command Set Specific Identify Controller Data Structure (CNS 06h, CSI 00h)

Figure 120 defines the I/O Command Set specific Identify Controller data structure for the NVM Command Set.

Figure 120: I/O Command Set Specific Identify Controller Data Structure for the NVM Command Set

Bytes	O/M[1]	Description
00	O	**Verify Size Limit (VSL):** If the Verify Support (NVMVFYS) bit is set to '1' in the Optional NVM Command Support (ONCS) field in the Identify Controller data structure (refer to the Identify Controller data structure (CNS 01h) section in the NVM Express Base Specification), then: a) a non-zero value in this field indicates the recommended maximum data size for a Verify command (refer to section 3.3.5); and b) a value of 0h in this field indicates that no recommended maximum data size for a Verify command is reported. If the NVMVFYS bit is cleared to '0', then: a) a non-zero value in this field indicates that the controller supports the Verify command with the maximum data size limit indicated by this field (refer to section 3.3.5); and b) a value of 0h in this field indicates that the controller does not support the Verify command. The non-zero value is in units of the minimum memory page size (CAP.MPSMIN) and is reported as a power of two (2^n). If the MEM bit is cleared to '0' in the CTRATT field in the Identify Controller data structure, then this field includes the length of metadata, if metadata is interleaved with the logical block data. If the MEM bit is set to '1', then this field excludes the length of metadata.
01	O	**Write Zeroes Size Limit (WZSL):** If the Write Zeroes Support Variants (NVMWZSV) bit is set to '1' in the Optional NVM Command Support (ONCS) field, then: a) a non-zero value in this field indicates the recommended maximum data size for a Write Zeroes command (refer to section 3.3.8); and b) a value of 0h in this field indicates that no recommended maximum data size for a Write Zeroes command is reported. If the NVMWZSV bit is cleared to '0', then: a) a non-zero value in this field indicates that the controller supports the Write Zeroes command with the maximum data size limit indicated by this field (refer to section 3.3.8); and b) a value of 0h in this field indicates that the controller does not support the Write Zeroes command. If the MAXWZD bit in the ONCS field is set to '1', then the controller supports a larger maximum data size for Write Zeroes commands with the Deallocate bit set to '1' than the controller supports for Write Zeroes commands that have the Deallocate bit cleared to '0', and the controller shall: • set this field to a non-zero maximum data size value that applies to Write Zeroes commands with the Deallocate bit cleared to '0'; and • set the Write Zeroes with Deallocate Size Limit (WZDSL) field to a larger non-zero maximum data size value that applies to Write Zeroes commands with the Deallocate bit set to '1'. The non-zero value is in units of the minimum memory page size (CAP.MPSMIN) and is reported as a power of two (2^n). If the MEM bit is cleared to '0' in the CTRATT field in the Identify Controller data structure, then this field includes the length of metadata, if metadata is interleaved with the logical block data. If the MEM bit is set to '1', then this field excludes the length of metadata.

02	O	**Write Uncorrectable Size Limit (WUSL):** If the Write Uncorrectable Support Variants (NVMWUSV) bit is set to '1' in the Optional NVM Command Support (ONCS) field, then: a) a non-zero value in this field indicates the recommended maximum data size for a Write Uncorrectable command (refer to section 3.3.7); and b) a value of 0h in this field indicates that no recommended maximum data size for a Write Uncorrectable command is reported. If the NVMWUSV bit in the ONCS field is cleared to '0', then: a) a non-zero value in this field indicates that the controller supports the Write Uncorrectable command with the maximum data size limit indicated by this field (refer to section 3.3.7); and b) a value of 0h in this field indicates that the controller does not support the Write Uncorrectable command. The non-zero value is in units of the minimum memory page size (CAP.MPSMIN) and is reported as a power of two (2^n). If the MEM bit is cleared to '0' in the CTRATT field in the Identify Controller data structure, then this field includes the length of metadata, if metadata is interleaved with the logical block data. If the MEM bit is set to '1', then this field excludes the length of metadata.
03	O	**Dataset Management Ranges Limit (DMRL):** If the Dataset Management Support Variants (NVMDSMSV) bit is set to '1' in the Optional NVM Command Support (ONCS) field, then: a) a non-zero value in this field indicates the recommended maximum number of logical block ranges for a Dataset Management command (refer to section 3.3.3); and b) a value of 0h in this field indicates that no recommended maximum number of logical block ranges for a Dataset Management command is reported. If the NVMDSMSV bit in the ONCS field is cleared to '0', then: a) a non-zero value in this field indicates that the controller supports the Dataset Management command with the maximum number of logical block ranges limit indicated by this field (refer to section 3.3.3); and b) a value of 0h in this field indicates that the controller does not support the Dataset Management command.
07:04	O	**Dataset Management Range Size Limit (DMRSL):** If the Dataset Management Support Variants (NVMDSMSV) bit is set to '1' in the Optional NVM Command Support (ONCS) field, then: a) a non-zero value in this field indicates the recommended maximum number of logical blocks in a single range for a Dataset Management command (refer to section 3.3.3); and b) a value of 0h in this field indicates that no recommended maximum number of logical blocks in a single range for a Dataset Management command is reported. If the NVMDSMSV bit in the ONCS field is cleared to '0', then: a) a non-zero value in this field indicates that the controller supports the Dataset Management command with the maximum number of logical blocks in a single range limit indicated by this field (refer to section 3.3.3); and b) a value of 0h in this field indicates that the controller does not support the Dataset Management command.

<table>
<tr><td>15:08</td><td>O</td><td>Dataset Management Size Limit (DMSL): If the Dataset Management Support Variants (NVMDSMSV) bit is set to ‘1’ in the Optional NVM Command Support (ONCS) field, then:
a) a non-zero value in this field indicates the recommended maximum total number of logical blocks for a Dataset Management command (refer to section 3.3.3).
b) a value of 0h in this field indicates that no recommended maximum total number of logical blocks for a Dataset Management command is reported.
If the NVMDSMSV bit in the ONCS field is cleared to ‘0’, then:
a) a non-zero value in this field indicates that the controller supports the Dataset Management command with the maximum total number of logical blocks limit indicated by this field (refer to section 3.3.3); and
b) a value of 0h in this field indicates that the controller does not support the Dataset Management command.</td></tr>
<tr><td>16</td><td>O</td><td>Key Per I/O Capabilities (KPIOCAP): This field indicates the attributes for Key Per I/O capability (refer to section 5.4).
<table><tr><th>Bits</th><th>Description</th></tr><tr><td>7:2</td><td>Reserved</td></tr><tr><td>1</td><td>Key Per I/O Scope (KPIOSC): If this bit is set to ‘1’, then the Key Per I/O capability applies to all namespaces in the NVM subsystem when Key Per I/O capability is enabled. If this bit is cleared to ‘0’, then the Key Per I/O capability does not apply to all namespaces in the NVM subsystem and is allowed to be independently enabled and disabled uniquely on each namespace within the NVM subsystem.</td></tr><tr><td>0</td><td>Key Per I/O Supported (KPIOS): If this bit is set to ‘1’, then the controller supports the Key Per I/O capability. If this bit is cleared to ‘0’, then the controller does not support the Key Per I/O capability.</td></tr></table></td></tr>
<tr><td>17</td><td>O</td><td>Write Zeroes With Deallocate Size Limit (WZDSL): A non-zero value in this field indicates the maximum data size for Write Zeroes commands with the Deallocate bit set to '1'. A 0h value in this field indicates that the maximum data size for Write Zeroes commands does not depend on the value of the Deallocate bit.
For Write Zeroes commands with the Deallocate bit cleared to ‘0’, this field has no effect (refer to the WZSL field).
For Write Zeroes commands with the Deallocate bit set to ‘1’, if this field is set to a non-zero value, then:
a) if the Write Zeroes Support Variants (NVMWZSV) bit in the ONCS field is set to ‘1’, then the value in this field is the recommended maximum data size and the value in the WZSL field is not used; and
b) if the NVMWZSV bit is cleared to ‘0’, then the value in this field is the maximum data size limit and the value in the WZSL field is not used.
If the WZSL field is cleared to 0h, then this field shall be cleared to 0h. If the WZSL field is not cleared to 0h, then the value of this field shall either be 0h or greater than the value in the WZSL field.
The non-zero value is in units of the minimum memory page size (CAP.MPSMIN) and is reported as a power of two (2^n).
If the MEM bit is cleared to ‘0’ in the CTRATT field in the Identify Controller data structure, then this field includes the length of metadata, if metadata is interleaved with the logical block data.
If the MEM bit is set to ‘1’, then this field excludes the length of metadata.</td></tr>
</table>

<table>
<tr><td>19:18</td><td>M</td><td>Admin Optional Command Support (AOCS): This field indicates the optional Admin commands and features supported by the controller.
<table><tr><th>Bits</th><th>Description</th></tr><tr><td>15:01</td><td>Reserved</td></tr><tr><td>00</td><td>Reporting Allocated LBA Supported (RALBAS): If this bit is set to ‘1’, then the controller supports the Get LBA Status capability with the Action Type value of 02h (refer to section 4.2.1). If this bit is cleared to ‘0’, then the controller does not support the Get LBA Status capability with the Action Type value of 02h.</td></tr></table></td></tr>
<tr><td>23:20</td><td>M</td><td>Version (VER): This field contains a Specification Version Descriptor (refer to the NVM Express Base Specification) indicating the version of this specification supported by the controller, as defined in Figure 121.</td></tr>
<tr><td>24</td><td>O</td><td>LBA Migration Queue Format (LBAMQF): This field indicates the format supported by this controller for User Data Migration Queue entries. Refer to section 5.6.
<table><tr><th>Value</th><th>Definition</th><th>Reference</th></tr><tr><td>0h</td><td>LBA Migration Queue Entry Type 0.</td><td>Figure 177</td></tr><tr><td>1h to BFh</td><td>Reserved</td><td></td></tr><tr><td>C0h to FFh</td><td>Vendor Specific</td><td></td></tr></table></td></tr>
<tr><td>4095:25</td><td>M</td><td>Reserved</td></tr>
<tr><td colspan="3">Notes:
1. O/M definition: O = Optional, M = Mandatory.</td></tr>
</table>

Published versions of this specification and the values that shall be reported by compliant controllers are defined in Figure 121.

Figure 121: NVM Command Set Specification Version Descriptor Field Values

Specification Versions [1]	MJR Field	MNR Field	TER Field
1.0	1h	0h	0h
1.1	1h	1h	0h
Notes: 1. The specification version listed includes lettered versions (e.g., 1.0 includes 1.0, 1.0a, 1.0b, etc.).			

4.1.5.5 NVM Command Set Identify Namespace Data Structure (CNS 09h, CSI 00h)

An NVM Command Set Identify Namespace data structure (refer to Figure 114) is returned to the host for the Format Index specified by the CNS Specific Identifier field as defined in Figure 122. The returned NVM Command Set Identify Namespace data structure specifies fields that define capabilities used by a host to format or create a namespace. If the specified Format Index is valid (refer to section 5.5), then the controller shall return an NVM Command Set Identify Namespace data structure that:

- for fields in Figure 114 that indicate “Yes” in the Reported column, contain a value that is the same for all namespaces using the specified Format Index; and
- for fields in Figure 114 that indicate “No” in the Reported column, contain a value cleared to 0h.

Figure 122: Command Dword 11 - CNS Specific Identifiers

Bits	Description
15:0	**Format Index (FIDX)**: This field specifies the Format Index identifying the LBA Format for which capabilities are to be returned. Refer to section 5.5.

4.1.5.6 Identify I/O Command Set specific Namespace data structure (CNS 0Ah, CSI 00h)

An I/O Command Set specific Identify Namespace data structure for the NVM Command Set (refer to Figure 118) is returned to the host for the Format Index specified by the CNS Specific Identifier field as defined in Figure 122. The returned I/O Command Set specific Identify Namespace data structure for the NVM Command Set specifies fields that define capabilities used by a host to format or create a namespace. If the specified Format Index is valid (refer to section 5.5), then the controller shall return an I/O Command Set specific Identify Namespace data structure for the NVM Command Set that:

- for fields in Figure 118 that indicate “Yes” in the Reported column, contain a value that is the same for all namespaces using the specified Format Index; and
- for fields in Figure 118 that indicate “No” in the Reported column, contain a value cleared to 0h.

4.1.5.7 Identify Namespace data structure for an Allocated Namespace ID (CNS 11h)

An Identify Namespace data structure (refer to Figure 114) is returned to the host for the specified namespace if the value in the Namespace Identifier (NSID) field is an allocated NSID. If the value in the NSID field specifies an unallocated NSID, then the controller returns a zero filled data structure.

If the value in the NSID field specifies an invalid NSID, then the controller shall abort the command with a status code of Invalid Namespace or Format. If the NSID field is set to FFFFFFFFh, then the controller shall abort the command with a status code of Invalid Namespace or Format.

4.1.5.8 Namespace Granularity List (CNS 16h)

If the controller supports reporting of Namespace Granularity (refer to section 5.5), then a Namespace Granularity List (refer to Figure 123) is returned to the host for up to:

a) 16 namespace granularity descriptors (refer to Figure 124) if the LBA Format Extension Enable (LBAFEE) field is cleared to 0h in the Host Behavior Support feature (refer to the Host Behavior Support section in the NVM Express Base Specification); or
b) 64 namespace granularity descriptors if the LBAFEE field is set to 1h in the Host Behavior Support feature.

Figure 123: Namespace Granularity List

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><th colspan="2">Header</th></tr>
<tr><td>03:00</td><td>Namespace Granularity Attributes (NGA): This field indicates attributes of the Namespace Granularity List.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:1</td><td>Reserved</td></tr>
<tr><td>0</td><td>Granularity Descriptor Mapping (GDM): If this bit is set to ‘1’, then each valid namespace granularity descriptor applies to the LBA format having the same Format Index and the Number of Descriptors field shall be equal to the sum of the values represented by the Number of LBA Formats field and the Number of Unique Attribute LBA Formats field in the Identify Namespace data structure (refer to Figure 114 and section 5.5). If this bit is cleared to ‘0’, then NG Descriptor 0 shall apply to all LBA formats and the Number of Descriptors field shall be cleared to 0h.</td></tr>
</table></td></tr>
<tr><td>04</td><td>Number of Descriptors (ND): This field indicates the number of valid namespace granularity descriptors in the list. This is a 0’s based value.
The namespace granularity descriptors with an index greater than the value in this field shall be cleared to 0h.</td></tr>
<tr><td>31:05</td><td>Reserved</td></tr>
<tr><th colspan="2">Namespace Granularity Descriptor List</th></tr>
<tr><td>47:32</td><td>NG Descriptor 0 (NGD0): This field contains the first namespace granularity descriptor in the list. This namespace granularity descriptor applies to LBA formats as indicated by the Granularity Descriptor Mapping bit.</td></tr>
<tr><td>63:48</td><td>NG Descriptor 1 (NGD1): This field contains the second namespace granularity descriptor in the list. This namespace granularity descriptor applies to LBA Format 1.</td></tr>
<tr><td>…</td><td>…</td></tr>
<tr><td>1055:1040</td><td>NG Descriptor 63 (NGD63): This field contains the last namespace granularity descriptor in the list. This namespace granularity descriptor applies to LBA Format 63.</td></tr>
</table>

The format of the namespace granularity descriptor is defined in Figure 124.

Figure 124: Namespace Granularity Descriptor

Bytes	Description
07:00	**Namespace Size Granularity (NSG):** Indicates the preferred granularity of allocation of namespace size when a namespace is created. The value is in bytes. A value of 0h indicates that the namespace size granularity is not reported.
15:08	**Namespace Capacity Granularity (NCG):** Indicates the preferred granularity of allocation of namespace capacity when a namespace is created. The value is in bytes. A value of 0h indicates that the namespace capacity granularity is not reported.

4.1.5.9 I/O Command Set specific Identify Namespace data structure for an Allocated Namespace ID (CNS 1Bh)

An I/O Command Set specific Identify Namespace data structure for the NVM Command Set (refer to Figure 118) is returned to the host for the namespace specified by the value in the NSID field if the specified NSID is an allocated NSID. If the specified NSID is not an allocated NSID (e.g., unallocated NSID or invalid NSID), then the controller behaves as specified in the NVM Express Base Specification.

4.1.5.10 Command Set Index Usage for the NVM Command Set

The following sections provide an example on how a host uses the CSI value of 00h for accessing Identify Namespace data structures for a namespace associated with the NVM Command Set.

4.1.5.10.1 Determining the Identify Command Information Associated with a Namespace

For a host to determine the Identify Namespace data structures (refer to section 1.4.2.4) for a namespace associated with the NVM Command Set, the host is required to issue the following Identify commands in any order:

a) An Identify command with:

   a. the CNS field set to 08h; and
   b. the NSID field set to the NSID of the namespace,

   to access the I/O Command Set Independent Identify Namespace data structure (refer to the NVM Express Base Specification);

b) An identify command with:

   a. the CNS field set to 00h; and
   b. the NSID field set to the NSID of the namespace,

   to access the Identify Namespace data structure (refer to section 4.1.5.1);

c) An Identify command with:

   a. the CNS field set to 05h;
   b. the CSI field set to 00h; and
   c. the NSID field set to the NSID of the zoned namespace,

   to access the I/O Command Set specific Identify Namespace data structure for the NVM Command Set (refer to section 4.1.5.3).

4.1.5.10.2 Determining the Identify Command Information Associated with a Format Index

For a host to determine the Identify Namespace data structures associated with a specific Format Index (i.e., determining information about a namespace associated with the NVM Command Set prior to creating that namespace), the host is required to issue the following Identify commands in any order:

a) An Identify command with:

   a. the CNS field set to 08h; and
   b. the NSID field set to FFFFFFFFh,

   to access the I/O Command Set Independent Identify Namespace data structure (refer to the NVM Express Base Specification);

b) An Identify command with:

   a. The CNS field set to 09h;
   b. The CSI field set to 00h;
   c. The NSID field set to 0h; and
   d. The CNS Specific Identifier field set to the Format Index,

   to access the Identify Namespace data structure (refer to section 4.1.5.5);

c) An Identify command with:

   a. the CNS set to OAh;
   b. the CSI set to 00h;
   c. the NSID field set to 0h; and
   d. the CNS Specific Identifier field set to the Format Index,

   to access the I/O Command Set specific Identify Namespace data structure for the NVM Command Set (refer to section 4.1.5.6).

4.1.6 Namespace Management command

The Namespace Management command operates as defined in the NVM Express Base Specification.

The host specified namespace management fields are specific to the I/O Command Set. The data structure passed to the create operation for the NVM Command Set (CSI 00h) is defined in Figure 125. Fields that are reserved should be cleared to 0h by host software. After successful completion of a Namespace Management command with the create operation, the namespace is formatted with the specified attributes.

If the LBA Format Extension Enable (LBAFEE) field is not set to 1h in the Host Behavior Support feature (refer to the Host Behavior Support section in the NVM Express Base Specification), then a controller aborts a Namespace Management command with a status code of Invalid Namespace or Format that specifies to create a namespace that is formatted with (refer to section 5.3.1):

a) 16b Guard Protection Information with the STS field set to a non-zero value;
b) 32b Guard Protection Information; or
c) 64b Guard Protection Information.

Implementations may impose requirements on which bits are allowed to be masked in the Logical Block Storage Tag Mask field (refer to Figure 125). Those requirements are defined in the LBSTM field in Figure 118 and the Storage Tag Masking Level Attribute field in Figure 118. If any of the requirements specified in those two fields are not met, then the controller shall abort the command with a status code of Invalid Field in Command.

If Flexible Data Placement (refer to the NVM Express Base Specification) is enabled in the specified Endurance Group and the Select field is set to Create (i.e., 0h):

- The Placement Handle List allows the host to specify the Reclaim Unit Handle associated with each specified Placement Handle. The number of Placement Handles in the list is specified by the NPHNDLS field which is limited to a value less than or equal to the lesser value of:
  - the number of Reclaim Unit Handles supported by the FDP configuration (refer to the NVM Express Base Specification); and
  - 128.
- If the NPHNDLS field is cleared to 0h, then:

  - if no namespace exists that was created with a Namespace Management command that specified the NPHNDLS field cleared to 0h in the specified Endurance Group, then the controller shall select a Reclaim Unit Handle for the Placement Handle 0 Associated RUH field that is not utilized by any namespace in the same Endurance Group that was created with a Namespace Management command that specified a non-zero NPHNDLS field; and
  - if any namespace exists that was created with a Namespace Management command that specified the NPHNDLS field cleared to 0h in the specified Endurance Group, then the controller shall utilize the same Reclaim Unit Handle for the Placement Handle 0 Associated RUH field that is utilized by those namespaces in the same Endurance Group that were created with a Namespace Management command that specified the NPHNDLS field cleared to 0h.

- If:
  - the NPHNDLS field is cleared to 0h; and
  - all Reclaim Unit Handle Identifiers accessible to the namespace are allocated to namespaces created by Namespace Management command with the NPHNDLS field set to a non-zero value,

  then the controller shall abort the command with a status code of Invalid Placement Handle List.
- If:
  - the NPHNDLS field is non-zero; and
  - a Reclaim Unit Handle Identifier specified by the host is the same as the Reclaim Unit Handle Identifier selected by the controller due to an existing namespace being created by Namespace Management command with the NPHNDLS field cleared to 0h,

  then the controller shall abort the command with a status code of Invalid Placement Handle List.
- If the NPHNDLS field is greater than the lesser value of:
  - the number of Reclaim Unit Handles supported by the FDP configuration (refer to the NVM Express Base Specification); and
  - 128,

  then the controller shall abort the command with a status code of Invalid Placement Handle List.
- If a Reclaim Unit Handle Identifier value in any entry in the Placement Handle List is greater than or equal to the number of Reclaim Unit Handles supported by the FDP configuration for the Endurance Group (refer to the NVM Express Base Specification), then controller shall abort the command with a status code of Invalid Placement Handle List.
- If the same Reclaim Unit Handle Identifier value is in two or more entries in the Placement Handle List, then controller shall abort the command with a status code of Invalid Placement Handle List.
- Namespaces that exist in the specified Endurance Group that utilize the same (i.e., share) Reclaim Unit Handle shall have the same user data format (i.e., report the same Format Index). If a Reclaim Unit Handle specified in the Placement Handle List is utilized by another namespace and the Format Index for that namespace does not match the specified Format Index for the namespace to be created, then the controller shall abort the command with a status code of Invalid Format.

Figure 125: Namespace Management – Host Software Specified Fields

Bytes	Description	Host Specified
Fields that are a subset of the Identify Namespace data structure (refer to Figure 114)		
07:00	**Namespace Size (NSZE)**	Yes

Figure 125: Namespace Management – Host Software Specified Fields

Bytes	Description	Host Specified
15:08	**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
91:31	Reserved	
95:92	**ANA Group Identifier (ANAGRPID)** [1]	Yes
99:96	Reserved	
101:100	**NVM Set Identifier (NVMSETID)** [1, 3]	Yes
103:102	**Endurance Group Identifier (ENDGID)** [1]	Yes
383:104	Reserved	
Fields that are not a subset of the Identify Namespace data structure.		
391:384	**Logical Block Storage Tag Mask (LBSTM)**	Yes
393:392	**Number of Placement Handles** [2] **(NPHNDLS):** This field specifies the number of Placement Handles included in the Placement Handle List. If the Flexible Data Placement capability (refer to the NVM Express Base Specification) is not supported or not enabled in specified Endurance Group, then the controller shall ignore this field.	Yes
498:394	Reserved	
511:499	Reserved for I/O Command Sets that extend this specification. Refer to the applicable I/O Command Set specification (e.g., Zoned Namespace Command Set Specification).	
Placement Handle List		
513:512	**Placement Handle 0 Associated RUH**[2]**:** This field specifies the Reclaim Unit Handle Identifier to be associated with the Placement Handle value 0h, if any. This Reclaim Unit Handle Identifier is used by the controller for any write commands that do not specify the Data Placement Directive. If the Flexible Data Placement capability (refer to the NVM Express Base Specification) is not supported or not enabled in specified Endurance Group, then the controller shall ignore this field.	Yes
515:514	**Placement Handle 1 Associated RUH**[2]**:** This field specifies the Reclaim Unit Handle Identifier to be associated with the Placement Handle value 1h, if any. If the Flexible Data Placement capability (refer to the NVM Express Base Specification) is not supported or not enabled in specified Endurance Group, then the controller shall ignore this field.	Yes
…		
767:766	**Placement Handle 127 Associated RUH**[2]**:** This field specifies the Reclaim Unit Handle Identifier to be associated with the Placement Handle value 127, if any. If the Flexible Data Placement capability (refer to the NVM Express Base Specification) is not supported or not enabled in specified Endurance Group, then the controller shall ignore this field.	Yes

Figure 125: Namespace Management – Host Software Specified Fields

Bytes	Description	Host Specified
4096:768	Reserved	

Notes:
1. A value of 0h specifies that the controller determines the value to use (refer to the Namespace Management section of the NVM Express Base Specification). If the associated feature is not supported, then this field is ignored by the controller.
2. Refer to the Flexible Data Placement section in the NVM Express Base Specification for requirements and use of this field. These fields are reserved if Flexible Data Placement is disabled in the specified Endurance Group.
3. NVM Sets are not supported if FDP is enabled in the specified Endurance Group as defined in the NVM Express Base Specification.

4.1.7 Sanitize command

The Sanitize command operates as defined in the NVM Express Base Specification.

In addition to the requirements in the NVM Express Base Specification, the following NVM Command Set Admin commands (refer to Figure 126) are allowed if a sanitize operation is in progress:

Figure 126: Sanitize Operations – Admin Commands Allowed

Admin Command	Additional Restrictions
Get Log Page	The log pages listed below are allowed in addition to the log pages listed in the NVM Express Base Specification.

Log Pages	Additional Restrictions
Error Information	Return zeroes in the User Data field.

4.1.8 Track Send command

Upon posting the successful completion to a Track Send command (refer the NVM Express Base Specification) that specifies:

- the Log User Data Changes management operation;
- the Logging Action (LACT) bit set to ‘1’; and
- a Controller Data Queue Identifier for a created LBA Migration Queue,

then the controller shall post entries into that LBA Migration Queue that identifies the logical block modifications to the attached namespaces on the controller associated with the LBA Migration Queue (refer to section 5.6). Posting of these entries into the LBA Migration Queue continues until the logging is stopped as specified by the NVM Express Base specification.

If the Reporting Allocated LBA Supported (RALBAS) bit is set to ‘1’ (refer to Figure 120), then after that Track Send command completes successfully, any logical block modifications to the attached namespaces on the controller associated with the LBA Migration Queue (refer to section 5.6) that occur during the processing of that Track Send command are reported by a Get LBA Status command with the Action Type field set to 02h (i.e., Return Allocated LBAs). Those logical block modifications to the attached namespaces on the controller associated with the LBA Migration Queue (refer to section 5.6) that occur during the processing of that Track Send command may also be logged into the LBA Migration Queue.

If the RALBAS bit is cleared to ‘0’, then after that Track Send command completes successfully, any logical block modifications to the attached namespaces on the controller associated with the LBA Migration Queue (refer to section 5.6) that occur during the processing of that Track Send command may be logged into the LBA Migration Queue.

The posting of an entry into the LBA Migration Queue as a result of a Track Send command to indicate:

- the start of logging user data changes; or
- the stop of logging user data changes,

is not required to be posted prior to posting the completion of that Track Send command.

If:

- the specified LBA Migration Queue in Track Send command with the Log User Data Changes management operation and the Logging Action (LACT) bit set to '1'; and
- that specified LBA Migration Queue is currently full,

then the controller may abort the command with a status code of Controller Data Queue Full.

If the Track Send command is successful and there is a pending Controller Data Queue Full Error event for the LBA Migration Queue specified by the CDQID field, then the controller shall discard that pending event with no effects to the state of the LBA Migration Queue.

4.2 I/O Command Set Specific Admin commands

In addition to the I/O Command Set Specific Admin commands defined in the NVM Express Base Specification, the NVM Command Set defines the Admin commands defined in this section.

4.2.1 Get LBA Status command

The Get LBA Status command requests information about LBAs (refer to section 5.2.1). If the Get LBA Status command completes successfully, then the LBA Status Descriptor List, defined in Figure 131, is returned in the data buffer for that command.

The Get LBA Status 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.

The Maximum Number of Dwords (MNDW) field contains the maximum number of dwords to return. Upon successful command completion, the actual amount of data returned by the controller is indicated by the Number of LBA Status Descriptors (NLSD) field in the LBA Status Descriptor List.

Figure 127: Get LBA Status – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the start of the data buffer. Refer to the Common Command Format figure in the NVM Express Base Specification for the definition of this field.

Figure 128: Get LBA Status – Command Dword 10 and Command Dword 11

Bits	Description
63:00	**Starting LBA (SLBA):** This field indicates the 64-bit address of the first logical block addressed by this command. Command Dword 10 contains bits 31:00; Command Dword 11 contains bits 63:32.

Figure 129: Get LBA Status – Command Dword 12

Bits	Description
31:00	**Maximum Number of Dwords (MNDW):** This field specifies the maximum number of dwords to return. This is a 0's based value.

Figure 130: Get LBA Status – Command Dword 13

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:24</td><td>Action Type (ATYPE): This field specifies the mechanism the controller uses in determining the LBA Status Descriptors to return as defined in Figure 132.
<table>
<tr><th>Value</th><th>M/O/P[1]</th><th>Definition</th><th>Reference Section</th></tr>
<tr><td>02h</td><td>O</td><td>Return tracked allocated LBAs in the specified range</td><td>4.2.1.1.1</td></tr>
<tr><td>10h</td><td>O</td><td>Perform a scan and return Untracked LBAs and Tracked LBAs in the specified range</td><td rowspan="2">4.2.1.1.2</td></tr>
<tr><td>11h</td><td>O</td><td>Return Tracked LBAs in the specified range</td></tr>
<tr><td>All others</td><td></td><td>Reserved</td><td></td></tr>
<tr><td colspan="4">Notes:
1. O = Optional, M = Mandatory, P = Prohibited</td></tr>
</table></td></tr>
<tr><td>23:16</td><td>Reserved</td></tr>
<tr><td>15:00</td><td>Range Length (RL): This field specifies the length of the range of contiguous LBAs, beginning at Starting LBA (SLBA), that the action specified in the Action Type (ATYPE) field shall be performed on. A value of 0h in this field specifies the length of a range beginning at Starting LBA and ending at Namespace Size (NSZE) minus 1 (refer to Figure 114).</td></tr>
</table>

4.2.1.1 Get LBA Status Action Type Mechanisms

4.2.1.1.1 Return Allocated LBAs (Action Type 02h)

The Get LBA Status command specifying an Action Type of 02h requests information about allocated LBAs. The controller shall return information about allocated LBAs in the range specified in the Get LBA Status command for the namespace specified in the NSID field. A controller is allowed to track the Allocated LBAs on the alignment and granularity specified by the Tracked LBA Allocation Granularity (TLBAAG) field in the I/O Command Set specific Identify Namespace data structure for the NVM Command Set (refer to Figure 118).

If all the logical blocks within the same alignment and granularity unit are deallocated, then all LBAs in that granularity and alignment unit that are in the LBA range specified in the Get LBA Status command shall not be reported as being allocated in the returned information for the Get LBA Status command.

If all of the logical blocks within the same alignment and granularity unit are allocated, then all LBAs in that granularity and alignment unit that are in the LBA range specified in the Get LBA Status command shall be reported as being allocated in the returned information for the Get LBA Status command as being allocated.

If within the same alignment and granularity unit:

- one or more logical blocks are not deallocated; and
- one or more logical blocks are deallocated,

then all LBAs in that granularity and alignment unit that are in the LBA range specified in the Get LBA Status command shall be reported as being allocated in the returned information for the Get LBA Status command as being allocated.

4.2.1.1.2 Potentially Unrecoverable LBAs (Action Type 10h and 11h)

A Get LBA Status command specifying an Action Type of 10h or 11h requests information about Potentially Unrecoverable LBAs.

A controller identifies Potentially Unrecoverable LBAs using the following two report types:

a) **Tracked LBAs:** a list of Potentially Unrecoverable LBAs associated with physical storage. These may be discovered through a background scan where the controller examines the media in the background or discovered through other means. The Tracked LBA list is able to be returned without significant delay; or
b) **Untracked LBAs:** a list of Potentially Unrecoverable LBAs generated by a scan originated by a Get LBA Status command with the ATYPE field set to 10h. The controller scans internal data structures related to the specified range of LBAs to determine which LBAs are Potentially Unrecoverable LBAs. The controller may use this scan to determine which LBAs in which namespaces are affected by a component (e.g., die or channel) failure. Significant delays may be incurred during the processing of a Get LBA Status command with the ATYPE field set to 10h. After the controller discovers Untracked LBAs, those LBAs may or may not be added to the list of Tracked LBAs.

If the value in the Action Type (ATYPE) field is set to 10h, then:

a) the controller shall generate a list of Untracked LBAs as described in this section;
b) the controller shall return Untracked LBAs and Tracked LBAs in the range specified in the Get LBA Status command for the specified namespace;
c) the controller shall remove all LBAs in the range specified in the Get LBA Status command, which prior to processing the Get LBA Status command were successfully re-written, from relevant internal data structures (e.g., internal Tracked LBA list);
d) the controller shall ensure that any such successfully re-written logical blocks are not reported in any LBA Status Descriptor Entries returned by the Get LBA Status command unless, after having been removed from relevant internal data structures and prior to processing the Get LBA Status command, those LBAs were newly detected as being Potentially Unrecoverable LBAs; and
e) the list of Untracked LBAs returned by the Get LBA Status command may be discarded by the controller or added to the Tracked LBA list once the command has completed.

If the value in the Action Type (ATYPE) field is set to 11h, then the controller shall:

a) return Tracked LBAs in the range specified in the Get LBA Status command for the specified namespace;
b) remove all LBAs in the range specified in the Get LBA Status command, which prior to processing the Get LBA Status command were successfully re-written, from relevant internal data structures (e.g., internal Tracked LBA list);
c) ensure that any such successfully re-written logical blocks are not reported in any LBA Status Descriptor Entries returned by the Get LBA Status command unless, after having been removed from relevant internal data structures and prior to processing the Get LBA Status command, those LBAs were newly detected as being Potentially Unrecoverable LBAs; and
d) not perform a foreground scan to generate and return Untracked LBAs.

In response to a Get LBA Status command, the controller returns LBA Status Descriptors that describe LBAs written by a Write Uncorrectable command in addition to any other LBAs that may return an Unrecovered Read Error status discovered through other mechanisms. The list of Tracked LBAs and the list of Untracked LBAs may be included in LBA Status Descriptor Entries that describe LBAs written by a Write Uncorrectable command. If an LBA Status Descriptor Entry describes only LBAs written by a Write Uncorrectable command, then the LBA Range Status (LBARS) field in the Status field should be set to 011b in that entry.

4.2.1.2 LBA Status Descriptor List

Figure 131 defines the LBA Status Descriptor List returned in the data buffer.

Figure 131: LBA Status Descriptor List

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><th colspan="2">Header</th></tr>
<tr><td>03:00</td><td>Number of LBA Status Descriptors (NLSD): This field indicates the number of LBA Status Descriptor Entries returned by the controller in this data structure. A value of 0h in this field indicates that no LBA Status Descriptor Entry list is returned.</td></tr>
<tr><td>04</td><td>Completion Condition (CMPC): This field indicates the condition that caused completion of the Get LBA Status command.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>0h</td><td>No indication of the completion condition.</td></tr>
<tr><td>1h</td><td>INCOMPLETE: The command completed as a result of transferring the number of Dwords specified in the MNDW field and:
for any ATYPE, additional LBA Status Descriptor Entries are available to transfer that are associated with the specified LBA range; or
for ATYPE set to 10h, the scan did not complete.</td></tr>
<tr><td>2h</td><td>COMPLETE: The command completed as a result of completing the action specified in the Action Type field over the number of logical blocks specified in the Range Length field and there are no additional LBA Status Descriptor Entries available to transfer that are associated with the specified range.</td></tr>
<tr><td>All others</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>07:05</td><td>Reserved</td></tr>
<tr><th colspan="2">LBA Status Descriptor Entry List</th></tr>
<tr><td>23:08</td><td>LBA Status Descriptor Entry 0: This field contains the first LBA Status Descriptor Entry in the list, if present.</td></tr>
<tr><td>39:24</td><td>LBA Status Descriptor Entry 1: This field contains the second LBA Status Descriptor Entry in the list, if present.</td></tr>
<tr><td>…</td><td>…</td></tr>
<tr><td>(NLSD*16+23): (NLSD*16+8)</td><td>LBA Status Descriptor Entry NLSD-1: This field contains the last LBA Status Descriptor Entry in the list, if present.</td></tr>
</table>

Figure 132: LBA Status Descriptor Entry

Bytes	Description
07:00	**Descriptor Starting LBA (DSLBA):** This field indicates the 64-bit address of the first logical block of the LBA range for which this LBA Status Descriptor Entry reports LBA status.
11:08	**Number of Logical Blocks (NLB):** This field indicates the number of contiguous logical blocks reported in this LBA Status Descriptor Entry. The controller should perform the action specified in the Action Type field in such a way that the value in this field reports the largest number of contiguous logical blocks possible (i.e., multiple consecutive LBA Status Descriptor Entries should not report contiguous LBAs that span those entries, but rather, LBA Status Descriptor Entries should be consolidated into the fewest number of LBA Status Descriptor Entries possible). This is a 0's based value.
12	Reserved

Figure 132: LBA Status Descriptor Entry

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>13</td><td>
Status (STAS): This field contains information about this LBA range.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:3</td><td>Reserved</td></tr>
<tr><td>2:0</td><td>
LBA Range Status (LBARS): This field indicates information about the logical blocks indicated in this LBA Status Descriptor Entry for the Action Type field values supported.
<table>
<tr><th>Value</th><th>Definition</th><th>Used with Action Type field Values</th></tr>
<tr><td>000b</td><td>Each logical block may:
report Unrecovered Read Error status as a result of media errors;
be a logical block for which the most recent write to the logical block was a Write Uncorrectable command; or
be read successfully.</td><td>10h and 11h</td></tr>
<tr><td>001b</td><td>Each logical block may:
report Unrecovered Read Error status as a result of media errors; or
be a logical block for which the most recent write to the logical block was a Write Uncorrectable command.</td><td>10h and 11h</td></tr>
<tr><td>010b</td><td>One or more of the reported logical blocks are allocated.</td><td>02h</td></tr>
<tr><td>011b</td><td>Each logical block is a:
logical block for which the most recent write to the logical block was a Write Uncorrectable command.</td><td>10h and 11h</td></tr>
<tr><td>100b to 111b</td><td>Reserved</td><td></td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>15:14</td><td>Reserved</td></tr>
</table>

The Descriptor Starting LBA (DSLBA) field in the first LBA Status Descriptor Entry returned in the LBA Status Descriptor List shall contain the lowest numbered LBA that is greater than or equal to the value specified in the Starting LBA field in the Get LBA Status command.

For subsequent LBA Status Descriptor Entries, the contents of the Descriptor Starting LBA field shall contain the value of the lowest numbered LBA meeting the requirements for the specified Action Type value that is greater than or equal to the sum of the values in:

a) the Descriptor Starting LBA field in the previous LBA Status Descriptor Entry; and
b) the Number of Logical Blocks field in the previous LBA Status Descriptor Entry.

4.2.1.3 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.

5 Extended Capabilities

5.1 Asymmetric Namespace Access Reporting

Asymmetric Namespace Access Reporting operates as defined in the NVM Express Base Specification with additional definitions specific to the NVM Command Set.

Figure 133 describes Asymmetric Namespace Access effects on command processing that are specific to the NVM Command Set and its associated Feature Identifiers.

Figure 133: ANA effects on NVM Command Set Command Processing

Command	ANA State	Effects on command processing
Get Features	ANA Inaccessible, ANA Persistent Loss, or ANA Change	The following NVM Command Set specific feature identifiers are not available[1]: a) Error Recovery (i.e., 05h).
Identify	ANA Inaccessible or ANA Persistent Loss	Capacity fields in the Identify Namespace data structure (refer to Figure 114) information are cleared to 0h.
Set Features	ANA Inaccessible	The saving of features shall not be supported and the following NVM Command Set specific feature identifiers are not available[1]: a) Error Recovery (i.e., 05h).
	ANA Change	The saving of features shall not be supported and the following NVM Command Set specific feature identifiers are not available[1]: a) Error Recovery (i.e., 05h).

Notes:
1. If the ANA state is ANA Inaccessible State, then commands that use feature identifiers or log pages that are not available shall abort with a status code of Asymmetric Access Inaccessible. If the ANA state is ANA Persistent Loss State, then commands that use feature identifiers or log pages that are not available shall abort with a status code of Asymmetric Access Persistent Loss. If the ANA state is ANA Change State, then commands that use feature identifiers or log pages that are not available shall abort with a status code of Asymmetric Access Transition.

5.2 Command Set Specific Capability

5.2.1 Get LBA Status

The Get LBA Status capability enables the host to obtain information about LBAs:

- Identifying LBAs that may be allocated in a namespace provides the host with the ability to minimize the user data accessed when copying that namespace for use cases such as namespace migration and snapshots. The Get LBA Status capability provides the host the ability to identify logical blocks that may be allocated in a namespace.
- Potentially Unrecoverable LBAs are LBAs that, when read, may result in the command that caused the media to be read being aborted with a status code of Unrecovered Read Error. The Get LBA Status capability provides the host with the ability to identify Potentially Unrecoverable LBAs. The logical block data and metadata, if any, are able to be recovered from another location and re-written.

If the Get LBA Status capability is supported, then the controller shall support the Get LBA Status command.

If Action Types 10h and 11h are supported (refer to the Get LBA Status Supported (GLSS) bit in the Optional Admin Command Support (OACS) field in the Identify Controller data structure), then the controller shall:

- support LBA Status Information Alert Notices (refer to the Optional Asynchronous Events Supported (OAES) field in the Identify Controller data structure);
- support the LBA Status Information log page;
- support the Log Page Offset and extended Number of Dwords (i.e., 32 bits rather than 12 bits) in the Get Log Page command (refer to the Attributes field of the Identify Controller data structure); and
- support the LBA Status Information Attributes Feature.

Prior to using a Get LBA Status command with the Action Type (ATYPE) field values 10h or 11h:

- The host should use the Get Features and Set Features commands with the LBA Status Information Attributes Feature (refer to section 4.1.3.5) to retrieve and optionally configure the LBA Status Information Report Interval; and
- If the host wishes to receive LBA Status Information Alert asynchronous events, the host should enable LBA Status Information Alert Notices (refer to Figure 98).

If LBA Status Information Alert Notices are enabled, the controller shall send an LBA Status Information Alert asynchronous event if:

a) there are Tracked LBAs and:

   a) the LBA Status Information Report Interval condition has been exceeded; or
   b) an implementation specific aggregate threshold, if any exists, of Tracked LBAs has been exceeded;

   or

b) a component (e.g., die or channel) failure has occurred that may result in the controller aborting commands with a status code of Unrecovered Read Error.

Upon receiving an LBA Status Information Alert asynchronous event, the host should send one or more Get Log Page commands for Log Page Identifier 0Eh with the Retain Asynchronous Event bit set to ‘1’ to read the LBA Status Information log page (refer to section 4.1.4.5).

Once the host has started reading the LBA Status Information log page with the Retain Asynchronous Event bit set to ‘1’, the controller shall not modify the contents of that log page until the host reads the LBA Status Information log page with the Retain Asynchronous Event bit cleared to ‘0’.

The LBA Status Information log page returns zero or more sets of per-namespace LBA Range Descriptors. Each LBA Range Descriptor specifies a range of LBAs that should be examined by the host in a subsequent Get LBA Status command (refer to section 4.2.1).

The Get LBA Status command requests information about Potentially Unrecoverable LBAs in a specified range.

The LBA Status Information Report Interval is restarted by the controller when the host issues a Get Log Page command for Log Page Identifier 0Eh with the Retain Asynchronous Event bit cleared to ‘0’. Issuing a Get Log Page command for Log Page Identifier 0Eh with the Retain Asynchronous Event bit cleared to ‘0’ causes an outstanding LBA Status Information Alert asynchronous event to be cleared if there is one outstanding on the controller.

When the host re-reads the header of the LBA Status Information log page with the Retain Asynchronous Event bit cleared to ‘0’, the host should ensure that the LBA Status Generation Counter matches the original value read. If these values do not match, there is newer LBA Status Information log page data available than the data returned the previous time the host read the LBA Status Information log page. In this case,

the host is not required to wait for the LBA Status Information Poll Interval (LSIPI) to pass before re-reading the LBA Status Information log page.

The host decides when to send Get LBA Status commands and when to recover the LBAs identified by the Get LBA Status commands, relative to when the host issues a Get Log Page command for Log Page Identifier 0Eh with the Retain Asynchronous Event bit cleared to '0'. Section 5.2.1.1 describes some example host implementations.

The Get LBA Status command may return zero or more LBA Status Descriptors (refer to Figure 131) for each LBA Range Descriptor (refer to Figure 111) returned by the LBA Status Information log page.

Figure 134: Example LBA Status Log Namespace Element returned by LBA Status Information Log Page

Bytes	Description	Value
03:00	**Namespace Element Identifier (NEID)**	1
07:04	**Number of LBA Range Descriptors (NLRD)**	2
08	**Recommended Action Type (RATYPE)**	11h (i.e., Tracked LBAs)
15:09	Reserved	0h
31:16	**LBA Range Descriptor 0:** This field contains the first LBA Range Descriptor in this LBA Status Log Namespace Element.	
47:32	**LBA Range Descriptor 1:** This field contains the second LBA Range Descriptor in this LBA Status Log Namespace Element.	

Bytes	Description	Value
07:00	**Range Starting LBA (RSLBA)**	10
11:08	**Range Number of Logical Blocks (RNLB)**	1,000
15:12	Reserved	0h

Bytes	Description	Value
07:00	**Range Starting LBA (RSLBA)**	15,000
11:08	**Range Number of Logical Blocks (RNLB)**	15,010
15:12	Reserved	0h

Figure 135: Example LBA Status Descriptors for Get LBA Status Command issued for LBA Range Descriptor 0 in Figure 134 (RSLBA set to 10, RNLB set to 1,000)

Bytes	Description	Value
03:00	**Number of LBA Status Descriptors (NLSD)**	3
04	**Completion Condition (CMPC)**	2
07:05	Reserved	0h
23:08	**LBA Status Descriptor Entry 0:** This field contains the first LBA Status Descriptor Entry in this list.	
39:24	**LBA Status Descriptor Entry 1:** This field contains the second LBA Status Descriptor Entry in this list.	
55:40	**LBA Range Descriptor 2:** This field contains the third LBA Status Descriptor Entry in this list.	

Bytes	Description	Value
07:00	**Descriptor Starting LBA (DSLBA)**	10
11:08	**Number of Logical Blocks (NLB)**	30
…	…	…

Bytes	Description	Value
07:00	**Descriptor Starting LBA (DSLBA)**	550
11:08	**Number of Logical Blocks (NLB)**	75
…	…	…

Bytes	Description	Value

Figure 135: Example LBA Status Descriptors for Get LBA Status Command issued for LBA Range Descriptor 0 in Figure 134 (RSLBA set to 10, RNLB set to 1,000)

Bytes	Description	Value		
		07:00	**Descriptor Starting LBA (DSLBA)**	1,000
		11:08	**Number of Logical Blocks (NLB)**	10
		…	…	…

Figure 136: Example LBA Status Descriptors for Get LBA Status Command issued for LBA Range Descriptor 1 in Figure 134 (RSLBA set to 15,000, RNLB set to 15,010)

<table>
<tr><th>Bytes</th><th>Description</th><th colspan="3">Value</th></tr>
<tr><td>03:00</td><td>Number of LBA Status Descriptors (NLSD)</td><td colspan="3">1</td></tr>
<tr><td>04</td><td>Completion Condition (CMPC)</td><td colspan="3">2</td></tr>
<tr><td>07:05</td><td>Reserved</td><td colspan="3">0h</td></tr>
<tr><td>23:08</td><td>LBA Status Descriptor Entry 0: This field contains the only LBA Status Descriptor Entry in this list.</td><td colspan="3"><table>
<tr><th>Bytes</th><th>Description</th><th>Value</th></tr>
<tr><td>07:00</td><td>Descriptor Starting LBA (DSLBA)</td><td>15,000</td></tr>
<tr><td>11:08</td><td>Number of Logical Blocks (NLB)</td><td>15,010</td></tr>
<tr><td>…</td><td>…</td><td>…</td></tr>
</table></td></tr>
</table>

5.2.1.1 Sample Get LBA Status Host Software Implementations (Informative)

5.2.1.1.1 Example Flow #1

1) Read the LBA Status Information log page with RAE bit set to ‘1’;
2) Complete all necessary Get LBA Status commands;
3) Complete all necessary recovery of the affected user data by rewriting that data; and
4) Read the LBA Status Information log page header with RAE bit cleared to ‘0’.

5.2.1.1.2 Example Flow #2

1) Read the LBA Status Information log page with RAE bit set to ‘1’;
2) Read the LBA Status Information log page with RAE bit cleared to ‘0’;
3) Issue some host-determined subset of the Get LBA Status commands indicated by the log page data;
4) Complete the recovery of the affected user data returned by the Get LBA Status commands issued so far;
5) Re-issue the Get LBA Status commands over the ranges associated with the re-written (i.e., recovered) user data;
6) Confirm that the re-written LBAs are no longer in the Tracked LBA List (if any are still there, they are there because they have been detected as newly bad again);
7) Add any new LBA ranges returned in the Get LBA Status commands to the list of LBAs still outstanding the host needs to recover; and
8) If the host has not processed all LBA ranges returned by:

    - the LBA Status Information log page in step 1; and
    - the Get LBA Status command(s) in step 7,

then go back to step 3.

5.2.2 Improving Performance through I/O Size and Alignment Adherence

I/O controllers may require constrained I/O sizes and alignments to achieve the full performance potential. There are several optional attributes that the controller uses to indicate these recommendations. If hosts do not follow these constraints, then the controller shall function correctly, but performance may be limited.

For best write performance, the host should issue Write command, Write Uncorrectable command, Write Zeroes command, and the write portion of the Copy command that specify:

a) a number of logical blocks that is:

    a. a multiple of the Namespace Preferred Write Granularity (NPWG) field (refer to Figure 114), if the NPWG field is defined; and
    b. a multiple of the number of logical blocks indicated by the Stream Write Size (SWS) field (refer to the Streams Directive – Return Parameters Data Structure figure in the NVM Express Base Specification), if the Streams Directive is enabled;

    and

b) a Starting LBA (SLBA) field that is aligned to the Namespace Preferred Write Alignment (NPWA) field (refer to Figure 114), if the NPWA field is defined.

Resolving conflicts between namespace attributes and Streams attributes is described in section 5.2.2.1.

The namespace preferred deallocate granularity is a number of logical blocks that is indicated by both the NPDG field (refer to Figure 114) and the NPDGL field. The NPDGL field is able to represent larger values than the NPDG field (refer to Figure 118). Support for these fields is indicated by the OPTPERF field (refer to Figure 114). If the NPDG field and the NPDGL field are both supported and indicate different values of namespace preferred deallocate granularity, then the host should use the value indicated by the NPDGL field.

The namespace preferred deallocate alignment is a number of logical blocks that is indicated by both the NPDA field (refer to Figure 114) and the NPDAL field (refer to Figure 118). The NPDAL field is able to represent larger values than the NPDA field. Support for these fields is indicated by the OPTPERF field (refer to Figure 114). If the NPDA field and the NPDAL field are both supported and indicate different non-zero values of namespace preferred deallocate alignment, then the host should use the value indicated by the NPDAL field. The NPDA field is a 0-based number (i.e., at least one is required to be defined) and the NPDAL field is a 1-based number (i.e., a value of 0h indicates not supported).

For best performance, the host should issue Dataset Management commands with the Attribute – Deallocate (AD) bit set to ‘1’ that specify:

a) a Length in Logical Blocks field that is a multiple of the namespace preferred deallocate granularity, if the namespace preferred deallocate granularity is defined; and
b) a Starting LBA field that is:

    a. a multiple of the namespace preferred deallocate alignment, if the namespace preferred deallocate alignment is defined; and
    b. a multiple of the Stream Granularity Size (SGS) field (refer to the Streams Directive – Return Parameters Data Structure figure in the NVM Express Base Specification) if the Streams Directive is enabled.

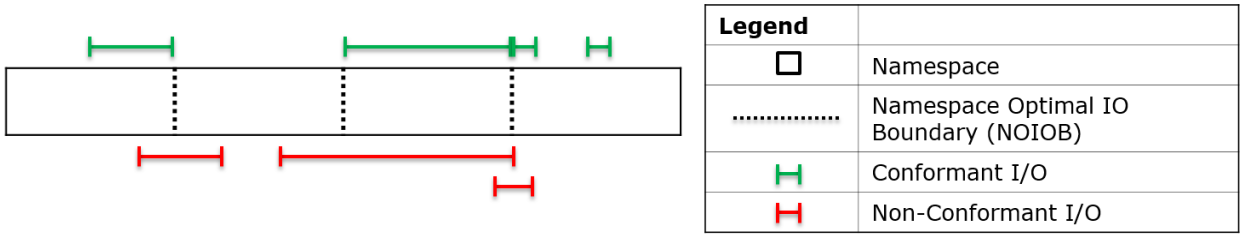
For best read performance, the host should issue Compare command, Read command, Verify command and the read portion of the Copy command that specify:

a) a number of logical blocks that is a multiple of the Namespace Preferred Read Granularity (NPRG) field (refer to Figure 118), if the NPRG field is supported; and
b) a Starting LBA (SLBA) field that is aligned to the Namespace Preferred Read Alignment (NPRA) field (refer to Figure 118), if the NPRA field is supported

5.2.2.1 Improved I/O examples (Informative)

It is recommended that the host utilize the I/O attributes as reported by the I/O controller to receive optimal performance from the NVM subsystem. This section summarizes performance related attributes from namespaces, streams, NVM Sets and the NVM command set. The I/O commands discussed throughout this section include those that interact with non-volatile medium in either a Read, Compare, Copy, Verify, Write, Write Uncorrectable, Write Zeroes operation, or Dataset Management operation with the Attribute - Deallocate bit set to '1'. The I/O command properties of length and alignment are discussed throughout this section.

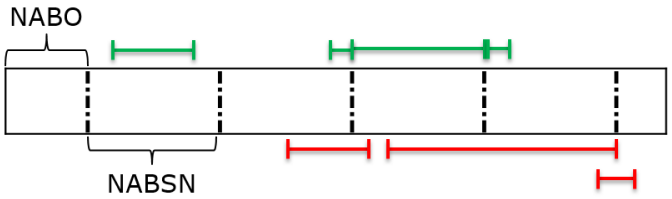
Figure 137: An example namespace with four NOIOBs



Legend	
	Namespace
	Namespace Optimal IO Boundary (NOIOB)
	Conformant I/O
	Non-Conformant I/O

In Figure 137 an example namespace is diagrammed with three Namespace I/O Boundaries (NOIOB) (refer to Figure 114). The NOIOB attribute should be applied to read and write commands as specified in section 5.2.2. An I/O command may see its performance limited if that I/O command does not conform to the NOIOB attribute considerations described in this section. The four green lines are example I/O commands from the host that adhere to the recommendations of NOIOB settings for this namespace. None of the four I/O commands shown in green on the top of Figure 137 cross an NOIOB. The three I/O commands shown in red on the bottom of Figure 137 violate the recommendations for improved performance. The longest I/O command shown in red crosses one NOIOB and ends aligned with a different NOIOB. The remaining two I/O commands shown in red also cross an NOIOB. All three of these example I/O commands shown in red are able to be split into two I/O commands that adhere to the recommendations provided by the namespace for NOIOB.

Figure 138: Example namespace illustrating a potential NABO and NABSN



Legend	
	Namespace
	Namespace Atomic Boundary Size Normal (NABSN)
	Conformant I/O
	Non-Conformant I/O
Namespace Atomic Boundary Offset (NABO)	

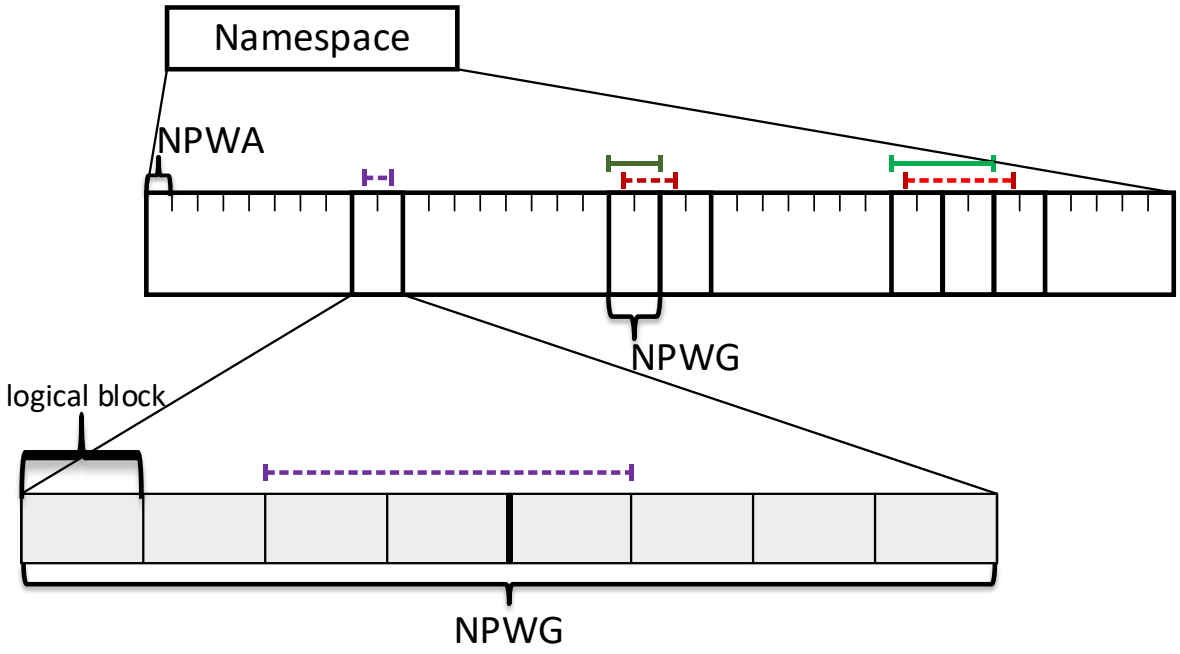
Continuing with the same namespace example from Figure 137, an illustration of Namespace Atomic Boundary Offset (NABO) (refer to Figure 114) and Namespace Atomic Boundary Size Normal (NABSN) (refer to Figure 114) is shown in Figure 138. NABSN and NABO attributes apply to Write, Write Uncorrectable, and Write Zeroes commands. NABSN and NOIOB may not be related to each other, and there may be an offset of NABO to locate the first NABSN starting. The NOIOBs are not shown in Figure 138. The I/O commands shown in green on the top of Figure 138 illustrate I/O commands that adhere to the namespace's guidance for optimal performance. The I/O commands shown in red on the bottom illustrate I/O commands that do not follow the optimal performance guidelines.

The I/O command examples shown in red in Figure 137 and Figure 138 both illustrate commands that are able to be restructured to conform to the namespace attributes for Optimal I/O relative to NOIOB, NABO, and NABSN. Each of these example I/O commands shown in red in Figure 137 and Figure 138 are able to be split into two different I/O commands that adhere to the recommendations. While this increases the number of commands sent to the controller, the expectation is that adherence to the boundary recommendations improves the performance for the controller. Avoiding host traffic that demands non-optimal I/O commands is the most recommendable solution for a host.

5.2.2.2 Alignment and Write Performance

NPWG and NPWA are namespace internal constructs, and they are illustrated in Figure 139. The box at the top of Figure 139 is the namespace. The series of boxes in the middle layer indicate many namespace optimal write units described by NPWA (refer to Figure 114) and NPWG (refer to Figure 114), and the bottom layer is a series of eight logical blocks that in aggregate form the NPWG for this example. Sometimes NPWG are useful because several sequential logical blocks (refer to Figure 114) may be placed and tracked together on the media, or the NPWG may be related to NVM subsystem data reliability implementation constraints.

Figure 139: Example namespace broken down to illustrate potential NPWA and NPWG settings



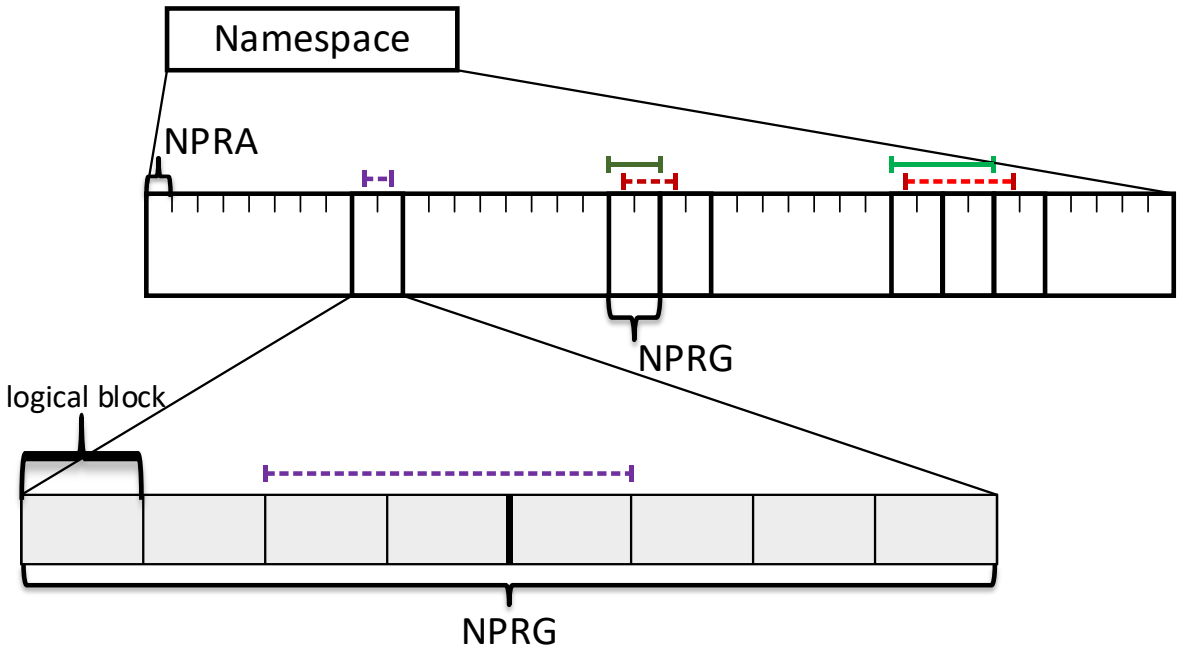
Legend	
	Conformant I/O
	Non-Conformant I/O
Namespace Preferred Write Alignment (NPWA)	
Namespace Preferred Write Granularity (NPWG)	

For a list of commands that apply to NPWG and NPWA attributes refer to section 5.2.2.

5.2.2.3 Alignment and Read Performance

NPRG and NPRA are namespace internal constructs, and they are illustrated in Figure 140. The box at the top of Figure 140 is the namespace. The series of boxes in the middle layer indicate many namespace optimal read units described by NPRA (refer to Figure 118) and NPRG (refer to Figure 118). In the figure, the read alignment is 4 logical blocks. The bottom layer is a series of eight logical blocks that in aggregate form the NPRG for this example. Sometimes NPRG are useful because several sequential logical blocks (refer to Figure 118) may be placed and tracked together on the media, or the NPRG may be related to NVM subsystem data reliability implementation constraints. Host reads that are of a length less than NPRA may see their performance impacted if they violate read alignment as described in Figure 118.

Figure 140: Example namespace broken down to illustrate potential NPRA and NPRG settings



Legend	
	Conformant I/O
	Non-Conformant I/O
Namespace Preferred Read Alignment (NPRA)	
Namespace Preferred Read Granularity (NPRG)	

The host should issue reads that meet the recommendations of NPRG and NPRA and may achieve optimal read performance by issuing reads that meet the recommendation of NORS. If NORS is greater than NPRG, reads that are a multiple of NPRG and not equal to NORS may see improved performance; however, read performance may not be optimal.

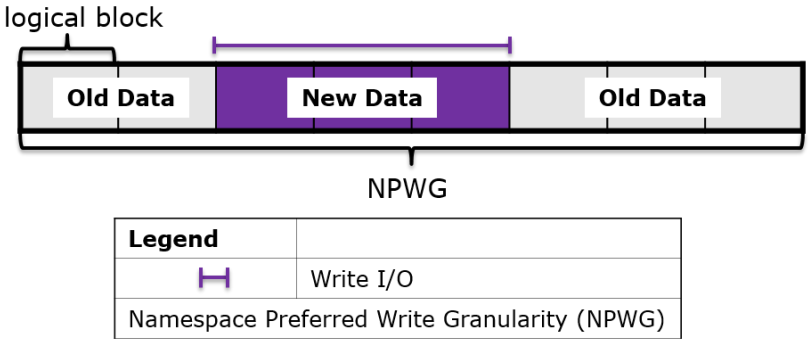
Non-adherence to write-related performance attributes (i.e., NPWG, NPWA, NPDG, NPDGL, NPDA, NPDAL, and NOWS), across all the namespaces in:

a) the same NVM Set;
b) the same Endurance Group when NVM Sets are not supported; or
c) the NVM subsystem when Endurance Groups are not supported,

may affect the level of read optimization achievable through the usage of NORS as described in this section.

For a list of commands that apply to NPRG and NPRA attributes refer to section 5.2.2.

Figure 141: Non-conformant Write Impact



Shown in Figure 141 is an I/O command that covers three of eight logical blocks within an NPWG. In this example namespace, NPWG is set to eight logical blocks, and the write of only three logical blocks requires a read of the preceding two logical blocks and trailing three logical blocks. The host write that completes to the non-volatile medium consists of five logical blocks of older data and three new logical blocks with the data provided by the write I/O command. The resulting read-modify-write may have non-optimal performance in comparison to a host write adhering to the NPWG attribute due to the extra read operation executed internally in the NVM subsystem. Aligning the beginning of the write I/O command with the NPWA attribute removes the requirement to read the preceding existing data. Host writes with a length that is a multiple of NPWG removes the requirement for reading the trailing data.

Following the NPWG recommendation alone is insufficient for optimal performance. If a write I/O command specifies the number of LBAs that is an integer multiple of NPWG and is offset in alignment from the recommended NPWA, then a read-modify-write may occur on the logical blocks at the beginning and ending of the command. The I/O commands shown in red in Figure 139 specify numbers of LBAs that are integer multiples of NPWG, but their alignment is triggering a read-modify-write at both the beginning and ending of the write I/O command. The write I/O commands shown in green adhere to the alignment and granularity requirements of the NPWA and NPWG. Figure 142 illustrates the shorter dark green write I/O command that adheres to both NPWG and NPWA attributes. This dark green write I/O command has a length equaling the NPWG attribute which adheres to the NPWG attribute recommendations. Figure 143 illustrates the dark red write I/O command that follows the NPWG attribute with a length of one NPWG, but that command does not adhere to the NPWA attribute recommendations. The dark red write I/O command requires a read of the old data at the beginning and the ending of the write I/O command to fill both NPWG units illustrated here. Longer write I/O commands that fail to adhere to the NPWA recommendation may trigger a read-modify-write of the leading and trailing NPWG segments inside of the NVM subsystem.

Figure 142: Host write I/O command following NPWA and NPWG

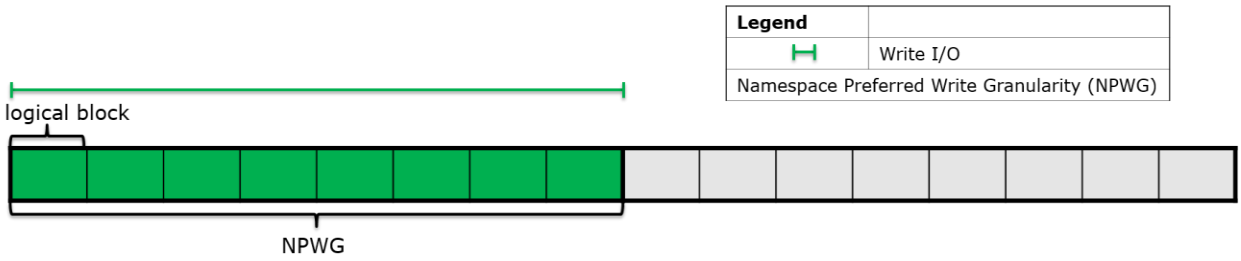
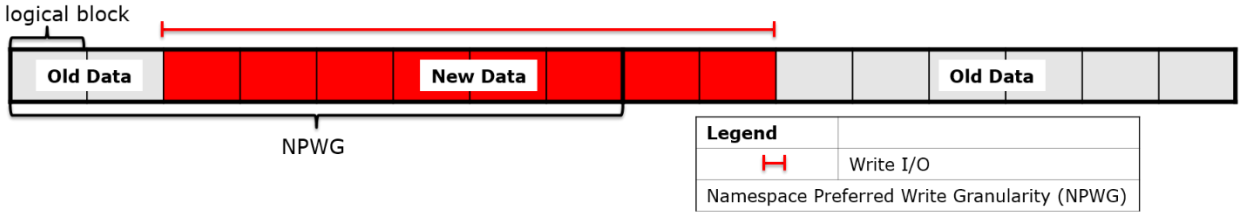
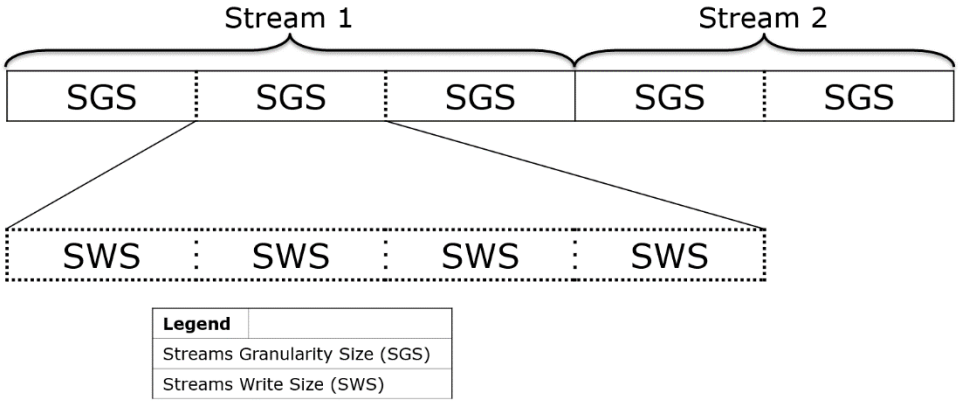


Figure 143: Host write I/O command following NPWG but not NPWA attributes



The namespace preferred deallocate granularity and the namespace preferred deallocate alignment (refer to section 5.2.2) are attributes of the namespace that are intended to improve performance for Dataset Management deallocate operations within a namespace. The namespace preferred deallocate granularity and the namespace preferred deallocate alignment may be impacted by multiple factors including but not limited to the boundaries described in Figure 139, device hardware limits, or non-volatile medium erase block sizes. Deallocating at multiples of the namespace preferred deallocate granularity and aligned to the namespace preferred deallocate alignment (i.e., (Starting LBA modulo namespace preferred deallocate alignment) == 0) may enable improved deallocate performance for the namespace.

Figure 144: Two streams composed of SGS and SWS



Streams (refer to the Streams Directive section in the NVM Express Base Specification) may or may not be utilized with different namespace attributes.

The stream granularity length is a number of logical blocks that is the product of the Stream Granularity Size (SGS) field and the number of logical blocks indicated by the Stream Write Size (SWS) field (refer to the Streams Directive – Return Parameters Data Structure figure in the NVM Express Base Specification). Figure 144 shows an example of the relationship between these attributes in which the SGS field is set to 4h. The first stream is composed of three SGS units, and each SGS unit in this example is equal to four SWS units. A stream is optimized for performance of the Dataset Management command deallocate operation if write I/O lengths are integer multiples of the stream granularity length. A stream is optimized for write performance if write I/O lengths are integer multiples of the SWS field.

Streams are sometimes handled by separate I/O paths in the device. This may include different device hardware, media mapping, or reliability protections. The number of logical blocks indicated by the SWS field should be a multiple of the number of logical blocks indicated by the NPWG field. The size indicated by the SGS field and the namespace preferred deallocate granularity may be equal to each other or multiples of each other. If a namespace indicates integer multiple size relationships between the streams

attributes (the SWS field and the SGS field) and the namespace attributes (the NPWG field and the namespace preferred deallocate granularity), then a write operation or a deallocate operation may obtain optimal performance by specifying a number of logical blocks that is equal to the largest of those attributes.

Not all namespaces indicate their Streams attributes and namespace attributes in multiples as described above. The recommended order of priority for a host to resolve conflicts between namespace attributes and Streams attributes is to issue write operations that conform to the SGS field and the SWS field if the Streams Directive is used. If the Streams Directive is not used, then issuing write operations that conform to the namespace attributes should provide improved performance.

If the Streams Directive is enabled on a namespace, and a Dataspace Management command specifying a deallocate operation specifies a range of logical blocks that are associated with a stream, then that range should conform to the SGS based alignment and size preferences. If the Streams Directive is not enabled on a namespace, or if the logical blocks specified by a range are not associated with a stream, then that range should conform to the namespace preferred deallocate granularity and the namespace preferred deallocate alignment.

Namespace Optimal Write Size (NOWS) (refer to Figure 114) is intended to supplement NVM Sets Optimal Write Size as NOWS provides a mechanism to report the optimal write size that scales to a multiple namespace per NVM Set use case, but also covers the use case where there is a single namespace that exists in an NVM Set. Namespaces should report NOWS as a multiple of NPWG. When constructing write operations, the host should minimally construct writes that meet the recommendations of NPWG and NPWA, but may achieve optimal write performance by constructing writes that meet the recommendation of NOWS.

If NVM Sets are supported as described in Figure 114, the value in the NOWS field for the namespace indicates the value the host should use to achieve optimal performance. If an NVM Set does not specify an Optimal Write Size, the host should use the value in the NOWS field for the namespace for I/O optimization purposes. Similarly, if NOWS is not defined for a namespace, the host should use the value in the Optimal Write Size field for the NVM Set associated with that namespace to achieve optimal performance.

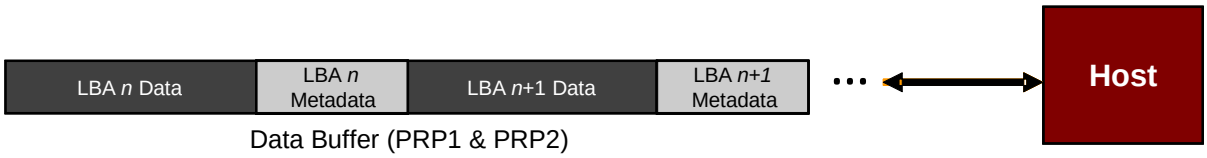
5.2.3 Metadata Handling

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.

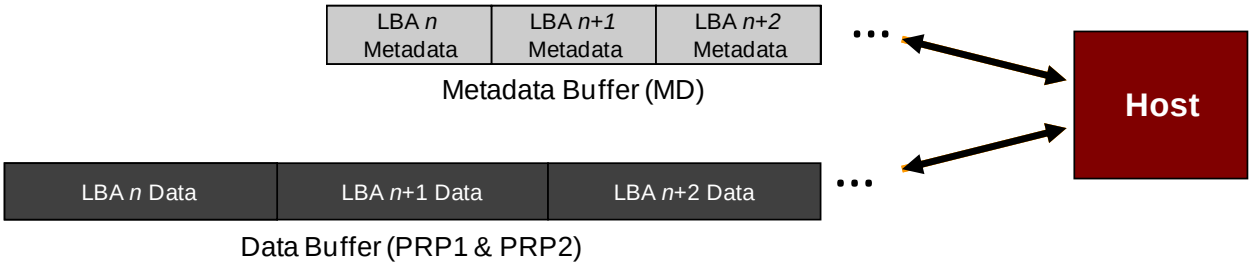
The first mechanism for transferring the metadata is as a contiguous part of the logical block that the metadata is associated with. The metadata is transferred at the end of the associated logical block, forming an extended logical block. This mechanism is illustrated in Figure 145. In this case, both 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 145: 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 146. 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 146: Metadata – Transferred as Separate Buffer



One of the transfer mechanisms shall be selected for each namespace when the namespace 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.

5.3 End-to-end Data Protection

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 the host to determine the integrity of the logical block. This additional protection information, if present, is either the first bytes of metadata or the last bytes of metadata, based on the format of the namespace (refer to the PIP bit in the DPS field shown in Figure 114). If the value in the Metadata Size (refer to Figure 116) is greater than the number of bytes of protection information and the protection information is contained in the first bytes of the metadata, then the CRC does not cover any metadata bytes. If the Metadata Size is greater than the number of bytes of protection information and the protection information is contained in the last bytes of the metadata, then the CRC covers all metadata bytes up to but excluding the protection information. As described in section 5.2.3, 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 145), while DIX functionality is achieved by configuring the metadata and data to be in separate buffers (as shown in Figure 146).

The NVM Express interface supports the same end-to-end protection types defined in the SCSI Protection information model specified in SBC-4. The type of end-to-end data protection (i.e., Type 1, Type 2, or Type 3) is selected when a namespace is formatted and is reported in the Identify Namespace data structure (refer to Figure 114).

5.3.1 Protection Information Formats

The following protection information formats are defined:

a) 16b Guard Protection Information;
b) 32b Guard Protection Information; and
c) 64b Guard Protection Information.

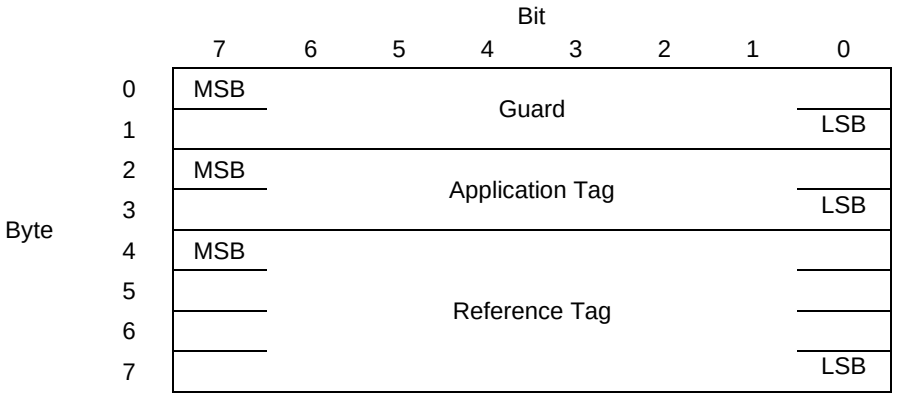
The following protection information formats are defined as extended protection information formats and are only supported when the Controller Attributes (CTRATT) field has the Extended LBA Formats Supported (ELBAS) bit set to ‘1’ (refer to the Identify Controller data structure in the NVM Express Base Specification):

a) 16b Guard Protection Information with the STS field set to a non-zero value;
b) 32b Guard Protection Information; and
c) 64b Guard Protection Information.

5.3.1.1 16b Guard Protection Information

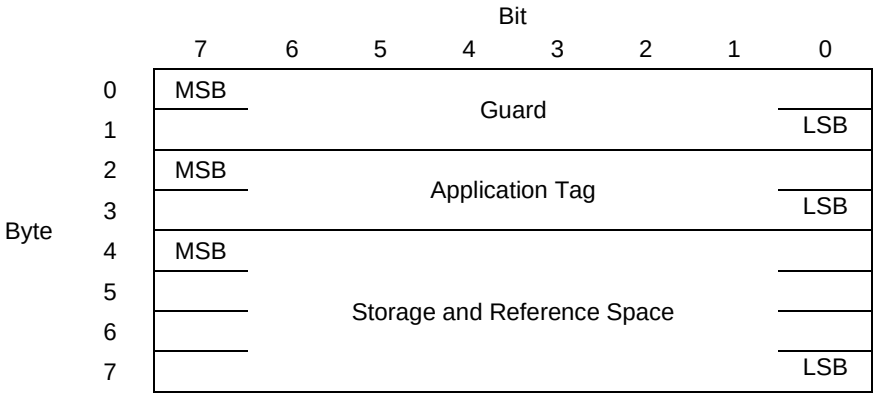
If the Storage Tag Size (STS) field for the LBA Format is cleared to 0h, then the 16b Guard Protection Information is shown in Figure 147 and is contained in the metadata associated with each logical block. The Guard field contains a CRC-16 computed over the logical block data. The formula used to calculate the CRC-16 is defined in SBC-4. In addition to a CRC-16, DIX also specifies an optional IP checksum that is not supported by the NVM Express interface. 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 147: 16b Guard Protection Information Format when STS field is cleared to 0h



If the Storage Tag Size (STS) field for the LBA Format is non-zero, then the 16b Guard Protection Information is shown in Figure 148. The Storage and Reference Space field is separated into a Storage Tag field and a Logical Block Reference Tag field as defined in section 5.3.1.4. The Storage Tag field is an opaque data field not interpreted by the controller. The Logical Block Reference Tag field associates logical block data with an address and protects against misdirected or out-of-order logical block transfer. The Logical Block Reference Tag field may be used to disable checking of protection information.

Figure 148: 16b Guard Protection Information Format when STS field is non-zero

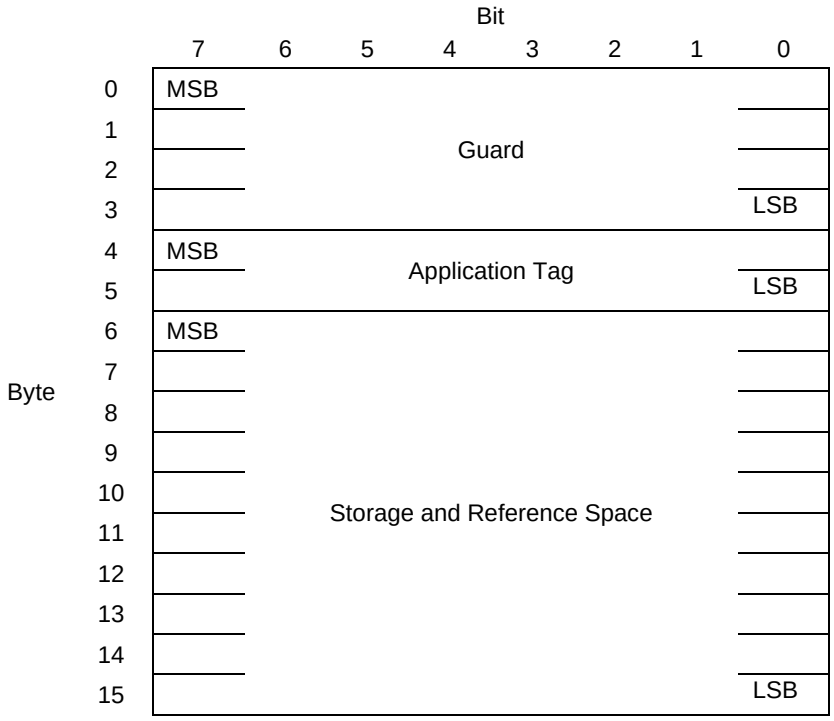


5.3.1.2 32b Guard Protection Information

The 32b Guard Protection Information is shown in Figure 149 and is contained in the metadata associated with each logical block. The 32b Guard Protection Information shall only be available to namespaces that have a logical block size (refer to the LBADS field in Figure 116) greater than or equal to 4 KiB.

The Guard field contains a 32b CRC computed over the logical block data. The formula used to calculate the CRC is the CRC-32C (Castagnoli) which uses the generator polynomial 1EDC6F41h (refer to the NVM Express Management Interface Specification). The Application Tag and Storage and Reference Space fields have the same definition as defined by 16b Guard Protection Information (refer to section 5.3.1.1).

Figure 149: 32b Guard Protection Information Format



5.3.1.2.1 32b CRC Test Cases

Several 32b CRC test cases are shown in Figure 150.

Figure 150: 32b CRC Test Cases for 4 KiB Logical Block with no Metadata

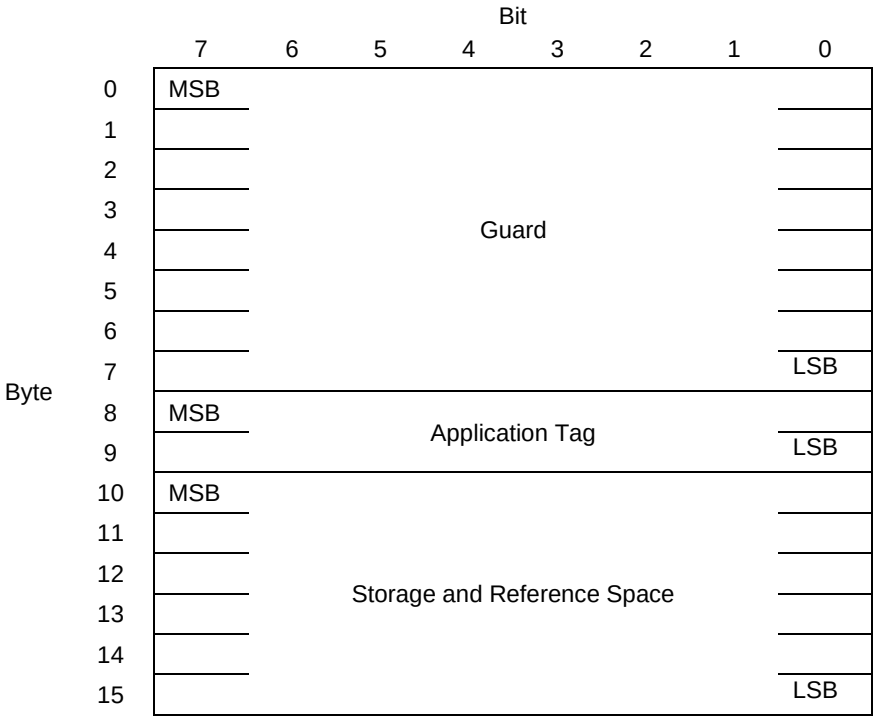
Logical Block Contents	32b Guard Field Value
4 KiB bytes each byte cleared to 00h	98F94189h
4 KiB bytes each byte set to FFh	25C1FE13h
4 KiB bytes of an incrementing byte pattern from 00h to FFh, repeating (e.g., byte 0 is set to 00h, byte 1 is set to 01h, … , byte 254 is set to FEh, byte 255 is set to FFh, byte 256 is set to 00h, …)	9C71FE32h
4 KiB bytes of a decrementing pattern from FFh to 00h, repeating (e.g., byte 0 is set to FFh, byte 1 is set to FEh, … , byte 254 is set to 01h, byte 255 is set to 00h, byte 256 is set to FFh, …)	214941A8h

5.3.1.3 64b Guard Protection Information

The 64b Guard Protection Information is shown in Figure 151 and is contained in the metadata associated with each logical block. 64b Guard Protection Information shall only be available to namespaces that have a logical block size (refer to the LBADS field in Figure 116) greater than or equal to 4 KiB.

The Guard field contains a 64b CRC computed over the logical block data. The polynomial used to calculate the CRC is defined in section 5.3.1.3.1. The Application Tag and Storage and Reference Space have the same definition as defined by 16b Guard Protection Information (refer to section 5.3.1.1).

Figure 151: 64b Guard Protection Information Format



5.3.1.3.1 64b CRC Definition

Figure 152 defines the 64b CRC polynomial used to generate the Guard field for the 64b Guard Protection Information.

Figure 152: 64b CRC Polynomials

Function	Definition
F(x)	A polynomial representing the transmitted logical block data, which is covered by the 64b CRC. For the purposes of the 64b CRC, the coefficient of the highest order term shall be byte zero bit seven of the logical block data and the coefficient of the lowest order term shall be bit zero of the last byte of the logical block data.
F'(x)	A polynomial representing the received logical block data.
G(x)	The generator polynomial: $G(x) = x^{64} + x^{63} + x^{61} + x^{59} + x^{58} + x^{56} + x^{55} + x^{52} + x^{49} + x^{48} + x^{47} + x^{46} + x^{44} + x^{41} + x^{37} + x^{36} + x^{34} + x^{32} + x^{31} + x^{28} + x^{26} + x^{23} + x^{22} + x^{19} + x^{16} + x^{13} + x^{12} + x^{10} + x^{9} + x^{6} + x^{4} + x^{3} + x^{0}$ (i.e., in finite field notation G(x) = 1_ AD93D235_94C93659h)
R(x)	The remainder polynomial calculated during CRC generation by the transmitter, representing the transmitted Guard field.
R'(x)	A polynomial representing the received Guard field.
RB(x)	The remainder polynomial calculated during CRC checking by the receiver. RB(x) = 0 indicates no error was detected.
RC(x)	The remainder polynomial calculated during CRC checking by the receiver. RC(x) = 0 indicates no error was detected.
QA(x)	The quotient polynomial calculated during CRC generation by the transmitter. The value of QA(x) is not used.

Figure 152: 64b CRC Polynomials

Function	Definition
QB(x)	The quotient polynomial calculated during CRC checking by the receiver. The value of QB(x) is not used.
QC(x)	The quotient polynomial calculated during CRC checking by the receiver. The value of QC(x) is not used.
M(x)	A polynomial representing the transmitted logical block data followed by the transmitted Guard field.
M'(x)	A polynomial representing the received logical block data followed by the received Guard field.

5.3.1.3.2 64b CRC Generation

The equations that are used to generate the 64b CRC from F(x) are as follows. All arithmetic is modulo 2.

The transmitter shall calculate the 64b CRC by appending 64 bits of zeroes to F(x) and dividing by G(x) to obtain the remainder R(x):

$$\frac{\left(x^{64} \times F(x)\right)}{G(x)} = QA(x) + \frac{R(x)}{G(x)}$$

R(x) is the 64b CRC value, and is transmitted in the Guard field.

M(x) is the polynomial representing the logical block data followed by the Guard field (i.e., F(x) followed by R(x)):

$$M(x) = \left(x^{64} \times F(x)\right) + R(x)$$

5.3.1.3.3 64b CRC Checking

M'(x) (i.e., the polynomial representing the received logical block data followed by the received Guard field) may differ from M(x) (i.e., the polynomial representing the transmitted logical block data followed by the transmitted Guard field) if there are transmission errors.

The receiver may check M'(x) validity by appending 64 bits of zeroes to F'(x) and dividing by G(x) and comparing the calculated remainder RB(x) to the received CRC value R'(x):

$$\frac{(x^{64} \times F'(x))}{G(x)} = QB(x) + \frac{RB(x)}{G(x)}$$

In the absence of errors in F'(x) and R'(x), the remainder RB(x) is equal to R'(x).

The receiver may check M'(x) validity by dividing M'(x) by G(x) and comparing the calculated remainder RC(x) to zero:

$$\frac{M'(x)}{G(x)} = QC(x) + \frac{RC(x)}{G(x)}$$

In the absence of errors in F'(x) and R'(x), the remainder RC(x) is equal to zero.

Both methods of checking M'(x) validity are mathematically equivalent.

5.3.1.3.4 Rocksoft™ Model CRC Algorithm parameters for 64b CRC

The 64-bit CRC required by this specification uses the generator polynomial AD93D235_94C93659h. The 64-bit CRC is calculated using the Rocksoft Model CRC Algorithm parameters defined in Figure 153.

Figure 153: 64-bit CRC Rocksoft Model Parameters

Parameter	Value
Name	“NVM Express 64b CRC”
Width	64
Poly	AD93D235_94C93659h
Init	FFFFFFFF_FFFFFFFFh
RefIn	True
RefOut	True
XorOut	FFFFFFFF_FFFFFFFFh
Check	11199E50_6128D175h

When sending a logical block and metadata, the 64b Guard field shall be calculated using the following procedure or a procedure that produces an equivalent result:

1. Initialize the CRC register to FFFFFFFF_FFFFFFFFh. This is equivalent to inverting the lowest 64 bits of the user data;
2. Append 64 bits of 0’s to the end of each logical block and metadata not including the 64b Protection Information. This results in the Guard field shown in Figure 151 to be cleared to 0h;
3. Map the bits from step 2 to the coefficients of the message polynomial M(x). Assume the length of M(x) is Y bytes. Bit 0 of byte 0 in the logical block is the most significant bit of M(x), followed by bit 1 of byte 0, on through to bit 7 of byte Y - 1. Note that the bits within each byte are reflected (i.e., bit n of each byte is mapped to bit (7 - n) resulting in bit 7 to bit 0, bit 6 to bit 1, and so on);

Figure 154: Logical Block and Metadata Example

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

4. Divide the polynomial M(x) by the generator polynomial AD93D235_94C93659h to produce the 64-bit remainder polynomial R(x);
5. Reflect each byte of R(x) (i.e., bit n of each byte is mapped to bit (7 - n) resulting in bit 7 to bit 0, bit 6 to bit 1, and so on) to produce the polynomial R′(x);
6. Invert R′(x) to produce the polynomial R″(x); and
7. Store R″(x) in the 64b Guard field in the 64b Protection Information.

Upon receipt of a logical block and metadata, the Guard field may be validated as follows:

1. Save the received Guard field;
2. Initialize the CRC register to FFFFFFFF_FFFFFFFFh. This is equivalent to inverting the lowest 64 bits of the logical block;
3. Clear the Guard field to 0h;
4. Map the bits in the logical block and metadata excluding the protection information to the coefficients of the message polynomial M(x) as described in step 3 in the Guard field calculation procedure for sending a logical block and metadata;
5. Divide the polynomial M(x) by the generator polynomial AD93D235_94C93659h to produce the 64-bit remainder polynomial R(x);
6. Reflect each byte of R(x) (i.e., bit n of each byte is mapped to bit (7 - n) resulting in bit 7 to bit 0, bit 6 to bit 1, and so on) to produce the polynomial R′(x);
7. Invert R′(x) to produce the polynomial R″(x); and
8. Compare R″(x) from step 7 to the Guard field value saved in step 1. If both values are equal, the 64b CRC check passes.

5.3.1.3.5 64b CRC Test Cases

Several 64b CRC test cases are shown in Figure 155.

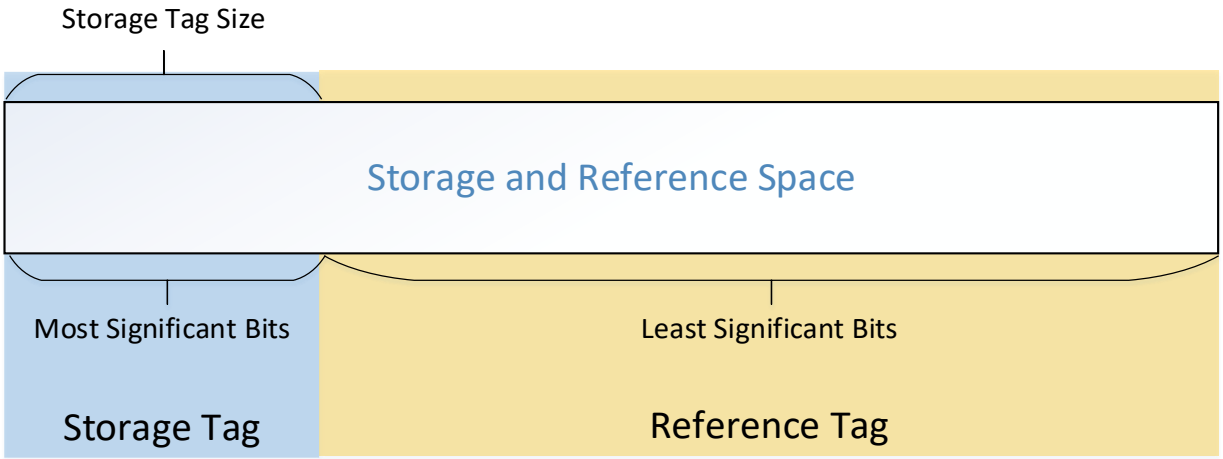
Figure 155: 64b CRC Test Cases for 4 KiB Logical Block with no Metadata

Logical Block Contents	64b Guard Field Value
4 KiB bytes each byte cleared to 00h	6482D367_EB22B64Eh
4 KiB bytes each byte set to FFh	C0DDBA73_02ECA3ACh
4 KiB bytes of an incrementing byte pattern from 00h to FFh, repeating (e.g., byte 0 is set to 00h, byte 1 is set to 01h, … , byte 254 is set to FEh, byte 255 is set to FFh, byte 256 is set to 00h, …)	3E729F5F_6750449Ch
4 KiB bytes of a decrementing pattern from FFh to 00h, repeating (e.g., byte 0 is set to FFh, byte 1 is set to FEh, … , byte 254 is set to 01h, byte 255 is set to 00h, byte 256 is set to FFh, …)	9A2DF64B8_E9E517Eh

5.3.1.4 Storage Tag and Logical Block Reference Tag from Storage and Reference Space

The Storage Tag Size (STS) field in the Identify Namespace data structure allows the separation of the Storage and Reference Space field in the protection information into a Storage Tag field and a Logical Block Reference Tag field as shown in Figure 156. If the STS field value is 0h, then no Storage Tag field is defined for the 16b Guard Protection Information and 64b Guard Protection Information formats. If the STS field value is non-zero, then that value specifies the number of most significant bits of the Storage and Reference Space field that is the Storage Tag field. The remaining least significant bits of the Storage and Reference Space field, if any, specify the Logical Block Reference Tag field. If the STS field value is equal to the size of the Storage and Reference Space field, then no Logical Block Reference Tag field is defined.

Figure 156: Separation of Storage and Reference Space into Storage Tag and Logical Block Reference Tag



5.3.1.4.1 Storage Tag Field and Logical Block Reference Tag Field

For I/O commands processed on namespaces with end-to-end protection enabled, the checking of the Storage Tag field, if defined, and the Logical Block Reference Tag requires variable sized Logical Block Storage Tag (LBST) field, Expected Logical Block Storage Tag (ELBST) field, Initial Logical Block Reference Tag (ILBRT) field, and Expected Initial Logical Block Reference Tag (EILBRT) field. This section defines the layout of these variable fields in Command Dword 2, Command Dword 3, and Command Dword

14. Figure 157 shows the minimum and maximum sizes of the LBST, ELBST, ILBRT and EILBRT fields based on the value of the Storage Tag Size (STS) field (refer to Figure 119) for each protection information format (refer to section 5.3.1).

Figure 157: LBST and LBRT Minimum and Maximum Sizes

STS Value	LBST/ELBST Bit Size	ILBRT/EILBRT Bit Size
16b Guard Protection Information		
0	0[1]	32
32	32	0[2]
32b Guard Protection Information		
16	16	64
64	64	16
64b Guard Protection Information		
0	0[1]	48
48	48	0[2]

Note:
1. Storage Tag field is not defined.
2. Logical Block Reference Tag field is not defined.

Figure 158 shows the layout of the LBST/ELBST and ILBRT/ EILBRT fields in Command Dword 2, Command Dword3, and Command Dword 14.

16b Guard Protection Information and 64b Guard Protection Information do not require the 80 bits allocated for the LBST, ELBST, ILBRT, and EILBRT fields in CDW 2, CDW 3, and CDW 14. Any unused bits are ignored by the controller (i.e., for 16b Guard Protection Information CDW2 and CDW 3 are ignored). If STS field value is 0h, then the Storage Tag field is not defined and the LBST and ELBST fields are not defined.

Figure 158: LBST, ELBST, ILBRT, and EILBRT fields Format in Command Dwords

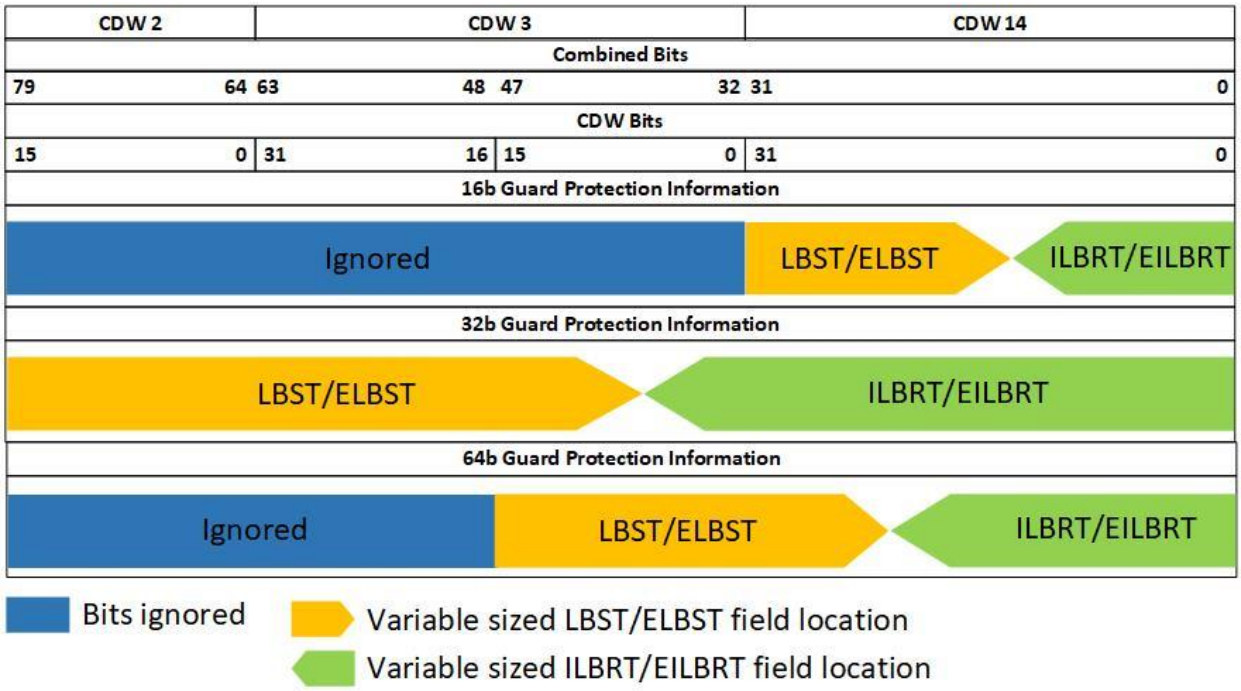


Figure 159 shows the layout of the LBST, ELBST, ILBRT, and EILBRT fields for I/O commands that utilize the fields.

Figure 159: I/O Command LBST, ELBST, ILBRT, and EILBRT fields Format

Bits	Description
16b Guard Protection Information	
Command Dword 2	
15:00	Ignored by the controller
Command Dword 3	
31:00	Ignored by the controller
Command Dword 14	
31:00	Variable sized LBST, ELBST, ILBRT or EILBRT as defined in Figure 158
32b Guard Protection Information	
Command Dword 2	
15:00	Most significant bit of the LBST or ELBST
Command Dword 3	
31:00	Variable sized LBST, ELBST, ILBRT or EILBRT as defined in Figure 158
Command Dword 14	
31:00	Variable sized LBST, ELBST, ILBRT or EILBRT as defined in Figure 158
64b Guard Protection Information	
Command Dword 2	
15:00	Ignored by the controller
Command Dword 3	
31:16	Ignored by the controller
15:00	Variable sized LBST, ELBST, ILBRT or EILBRT as defined in Figure 158
Command Dword 14	
31:00	Variable sized LBST, ELBST, ILBRT or EILBRT as defined in Figure 158

For an example of the 16b Guard Protection Information usage of Command Dword 2, Command Dword 3, and Command Dword 14 assume a namespace is formatted with the following:

a) LBA Data Size field (refer to Figure 116) cleared to 0h specifying a 512B logical block data size;
b) Metadata Size field (refer to Figure 116) set to 8h specifying an 8B metadata size;
c) Protection Information Format field (refer to Figure 118) cleared to 00b specifying the 16b Guard Protection Information; and
d) Storage Tag Size (STS) field (refer to Figure 119) cleared to 0h specifying that the Storage and Reference Space field is the Logical Block Reference Tag (i.e., the Storage Tag field is not defined),

then the definition of Command Dword 2, Command Dword 3, and Command Dword 14 for a Write command is shown in Figure 160 and for a Read command is shown in Figure 161.

Figure 160: 16b Guard Protection Information Write Command Example

Bits	Description
Command Dword 2	
15:00	Ignored by the controller
Command Dword 3	
31:00	Ignored by the controller
Command Dword 14	
31:00	ILBRT

Figure 161: 16b Guard Protection Information Read Command Example

Bits	Description
Command Dword 2	
15:00	Ignored by the controller
Command Dword 3	
31:00	Ignored by the controller
Command Dword 14	
31:00	EILBRT

For an example of the 32b Guard Protection Information usage of Command Dword 2, Command Dword 3, and Command Dword 14 assume a namespace is formatted with the following:

a) LBA Data Size field (refer to Figure 116) set to Ch specifying a 4 KiB logical block data size;
b) Metadata Size field (refer to Figure 116) set to 10h specifying a 16B metadata size;
c) Protection Information Format field (refer to Figure 118) set to 01b specifying the 32b Guard Protection Information; and
d) Storage Tag Size (STS) field (refer to Figure 119 set to 20h specifying that the most significant 32 bits of the Storage and Reference Space field are the Storage Tag field,

then the definition of Command Dword 2, Command Dword 3, and Command Dword 14 for a Write command is shown in Figure 162 and for a Read command is shown in Figure 163.

Figure 162: 32b Guard Protection Information Write Command Example

Bits	Description
Command Dword 2	
15:00	Most significant 16 bits of the LBST
Command Dword 3	
31:16	Least significant 16 bits of LBST
15:00	Most significant 16 bits of the ILBRT

Figure 162: 32b Guard Protection Information Write Command Example

Bits	Description
Command Dword 14	
31:00	Least significant 32 bits of ILBRT

Figure 163: 32b Guard Protection Information Read Command Example

Bits	Description
Command Dword 2	
15:00	Most significant 16 bits of the ELBST
Command Dword 3	
31:00	Least significant 16 bits of ELBST
15:00	Most significant 16 bits of EILBRT
Command Dword 14	
31:00	Least significant 32 bits of EILBRT

For an example of the 64b Guard Protection Information usage of Command Dword 2, Command Dword 3, and Command Dword 14 assume a namespace is formatted with the following:

a) LBA Data Size field (refer to Figure 116) set to Ch specifying a 4 KiB logical block data size;
b) Metadata Size field (refer to Figure 116) set to 10h specifying a 16B metadata size;
c) Protection Information Format field (refer to Figure 118) set to 10b specifying the 64b Guard Protection Information; and
d) Storage Tag Size (STS) field (refer to Figure 119) set to 12h specifying that the most significant 18 bits of the Storage and Reference Space field are the least significant 18 bits of the Storage Tag field,

then the definition of Command Dword 2, Command Dword 3, and Command Dword 14 for a Write command is shown in Figure 164 and for a Read command is shown in Figure 165.

Figure 164: 64b Guard Protection Information Write Command Example

Bits	Description
Command Dword 2	
15:00	Ignored by the controller
Command Dword 3	
31:16	Ignored by the controller
15:00	Most significant 16 bits of LBST
Command Dword 14	
31:30	Least significant 2 bits of LBST
29:00	ILBRT

Figure 165: 64b Guard Protection Information Read Command Example

Bits	Description
Command Dword 2	
15:00	Ignored by the controller
Command Dword 3	
31:16	Ignored by the controller
15:00	Most significant 16 bits of the ELBST
Command Dword 14	
31:30	Least significant 2 bits of ELBST
29:00	EILBRT

5.3.1.4.2 Host Considerations when Formatting with Storage Tag (Informative)

The Format NVM command does not provide a method to change the Storage Tag Size (STS) field of the LBA format that a namespace is formatted with:

- from the value 0h to a non-zero value; or
- from a non-zero value to a different non-zero value.

The Namespace Management command, if supported, is able to be used to delete the existing namespace and to create a namespace with a different combination of values in the Storage Tag Size (STS) field and the Logical Block Storage Tag Mask (LBSTM) field.

5.3.2 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.

5.3.2.1 Protection Information and Write Commands

Figure 166 provides some examples of the protection information processing that may occur as a side effect of a Write command.

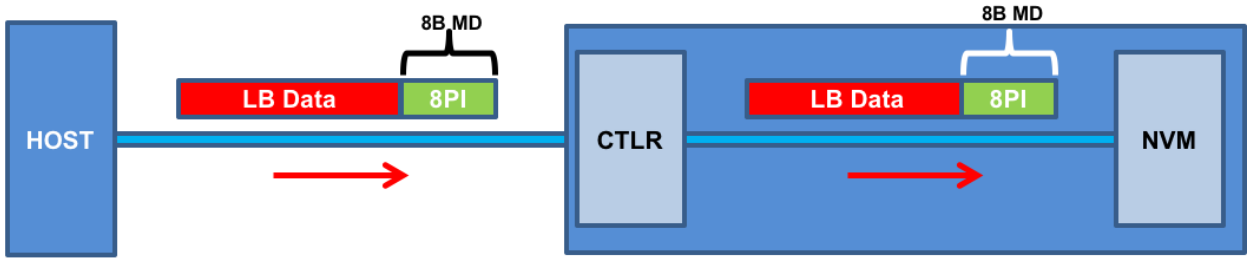
If the namespace is not formatted with end-to-end data protection, then logical block data and metadata is transferred from the host to the NVM with no protection information related processing by the controller.

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 Error, End-to-end Application Tag Check Error, End-to-end Storage Tag Check Error, or End-to-end Reference Tag Check Error).

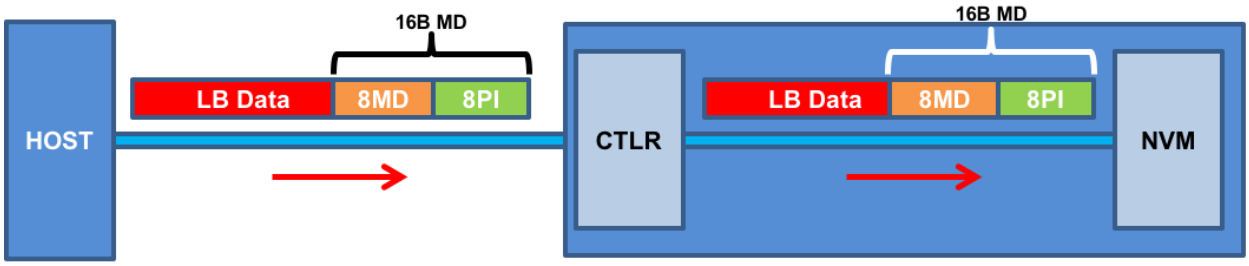
If the namespace is formatted with protection information and the PRACT bit is set to ‘1’, then:

1. If the namespace is formatted with Metadata Size (refer to Figure 116) equal to protection information size (refer to section 5.3.1), 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); and
2. If the namespace is formatted with Metadata Size greater than protection information size, 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 without checking the protection information portion regardless of PRCHK settings. 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).

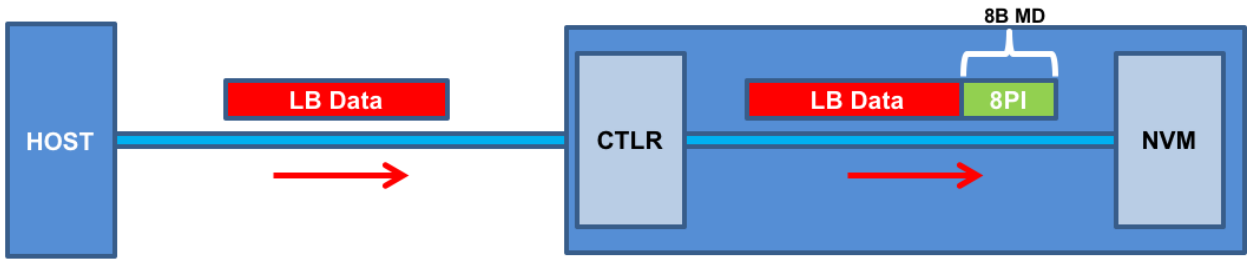
Figure 166: Write Command 16b Guard Protection Information Processing



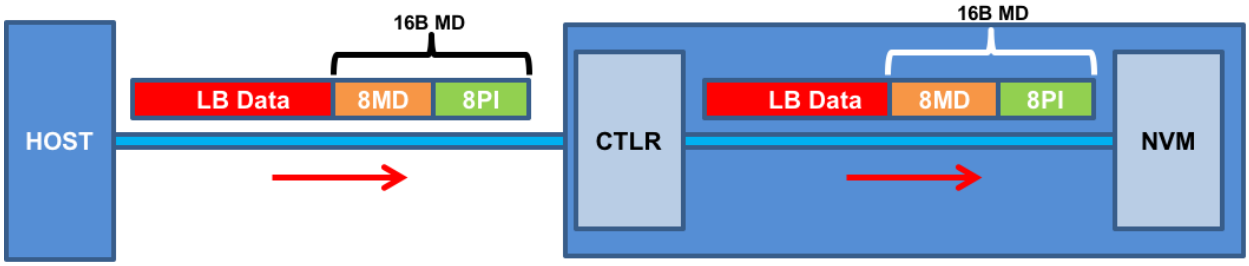
a) MD=8, PRACT=0: Metadata remains same size in NVM and host buffer



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

5.3.2.2 Protection Information and Read Commands

Figure 167 provides some examples of the protection information processing that may occur as a side effect of Read command processing.

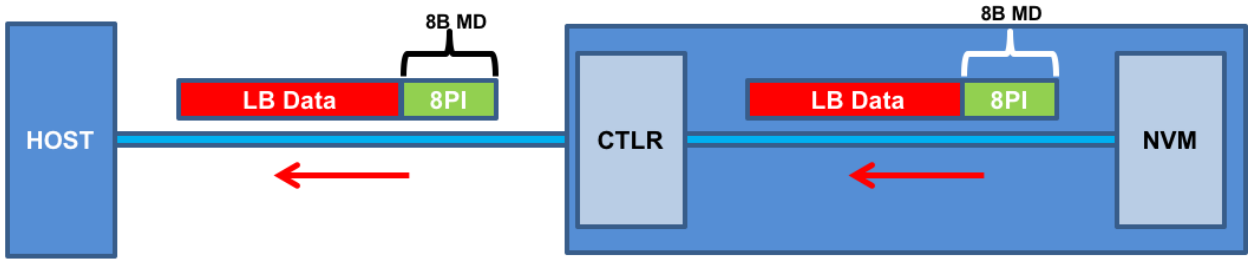
If the namespace is formatted with protection information and the PRACT bit is cleared to ‘0’, then the logical block data and metadata, which in this case contains the protection information and possibly

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 Error, End-to-end Application Tag Check Error, End-to-end Storage Tag Check Error, or End-to-end Reference Tag Check Error).

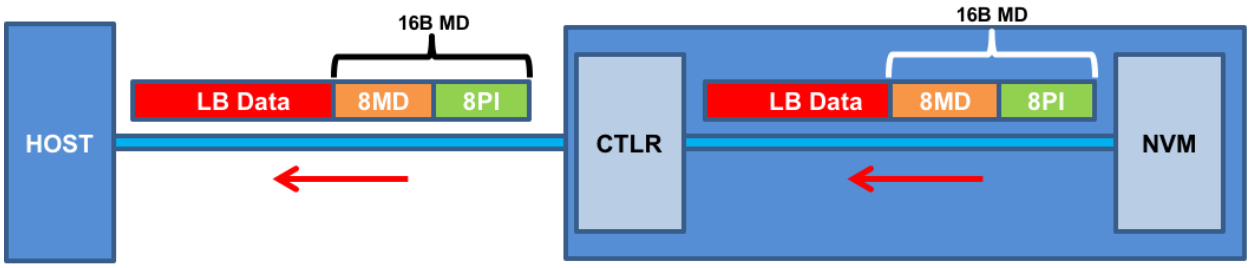
If the namespace is formatted with protection information and the PRACT bit is set to '1', then:

a) If the namespace is formatted with Metadata Size (refer to Figure 116) equal to protection information size (refer to section 5.3.1), 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 Error, End-to-end Application Tag Check Error, End-to-end Storage Tag Check Error, or End-to-end Reference Tag Check Error). After processing the protection information, the controller only returns the logical block data to the host (i.e., the metadata is not resident within the host buffer); and
b) if the namespace is formatted with Metadata Size greater than protection information size, 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 Error, End-to-end Application Tag Check Error, End-to-end Storage Tag Check Error, or End-to-end Reference Tag Check Error). 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).

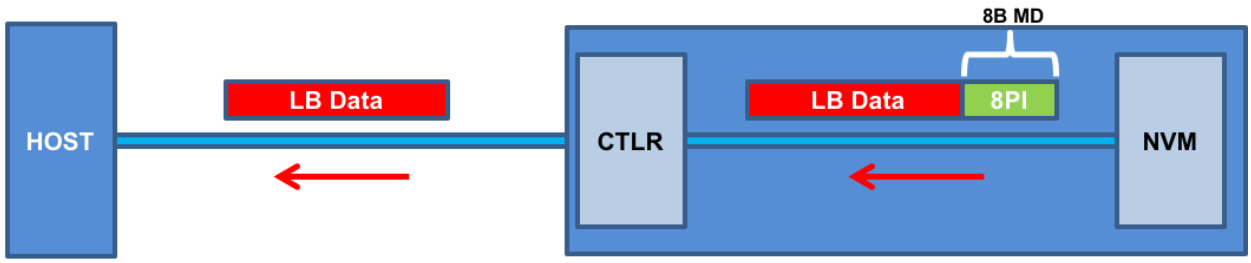
Figure 167: Read 16b Guard Command Protection Information Processing



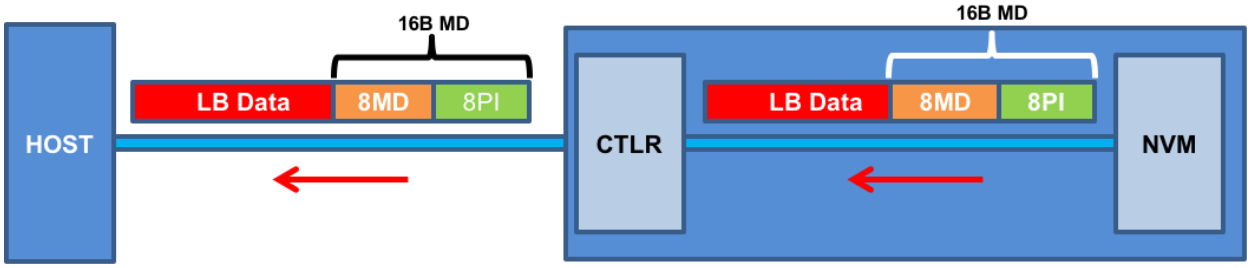
a) MD=8, PI, PRACT=0: Metadata remains same size in NVM and host buffer



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

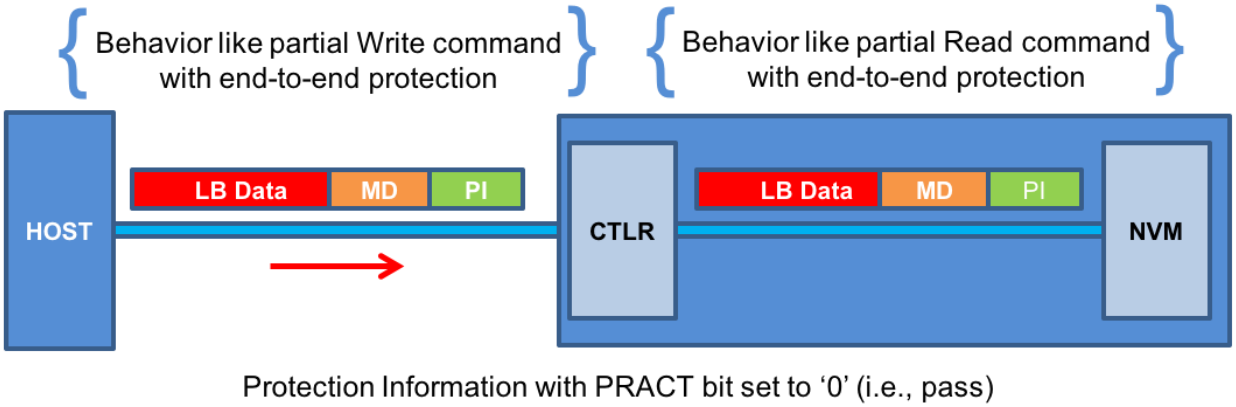
5.3.2.3 Protection Information for Fused Operations

Protection processing for fused operations is the same as those for the individual commands that make up the fused operation.

5.3.2.4 Protection Information and Compare commands

Figure 168 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. For the portion of the Compare command that transfers data and protection information from the host to the controller, the protection information checks performed by the controller parallels the Write command protection information checks (refer to section 5.3.2.1). For the portion of the Compare command that transfers data and protection information from the NVM media to the controller, the protection information checks performed by the controller parallels the Read command protection information checks (refer to section 5.3.2.2). The ELBST, EILBRT, PRINFO, STC, ELBATM, and ELBAT fields in the command are used by both sets of protection information checks as defined in section 5.3.3.

Figure 168: Protection Information Processing for Compare



5.3.2.5 Protection Information and Copy commands

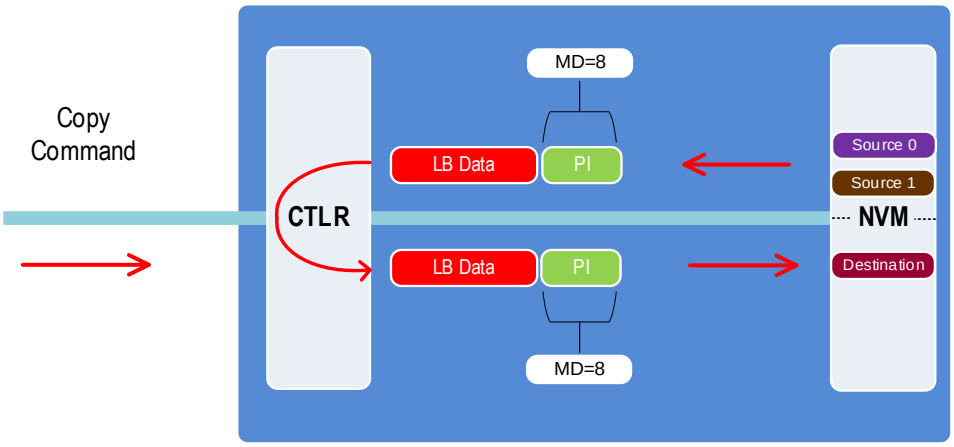
Protection information (PI) processing during a Copy command parallels both Write and Read commands. For the portion of the Copy command that transfers data and PI from the LBAs described by a Source Range entry (refer to Figure 39 and Figure 40), the PI checks performed by the controller are controlled by the PRINFOR field in Copy command Dword 12 (refer to Figure 34) and parallels the Read command PI checks (refer to section 5.3.2.2) as follows:

- The logical block data and metadata is transferred from the NVM to the controller.
- As the logical block data and metadata pass through the controller, the PI within the metadata is checked. If a PI check error is detected, the command completes with the status code of the error detected (i.e., End-to-end Guard Check Error, End-to-end Application Tag Check Error, End-to-end Storage Tag Check Error, or End-to-end Reference Tag Check Error).

For the portion of the Copy command that transfers data and PI to the LBAs starting at the SDLBA field (refer to Figure 33), the PI operations performed by the controller are controlled by the PRINFOW field in Copy command Dword 12 (refer to Figure 34) and parallels the Write command PI checks (refer to section 5.3.2.1) as follows:

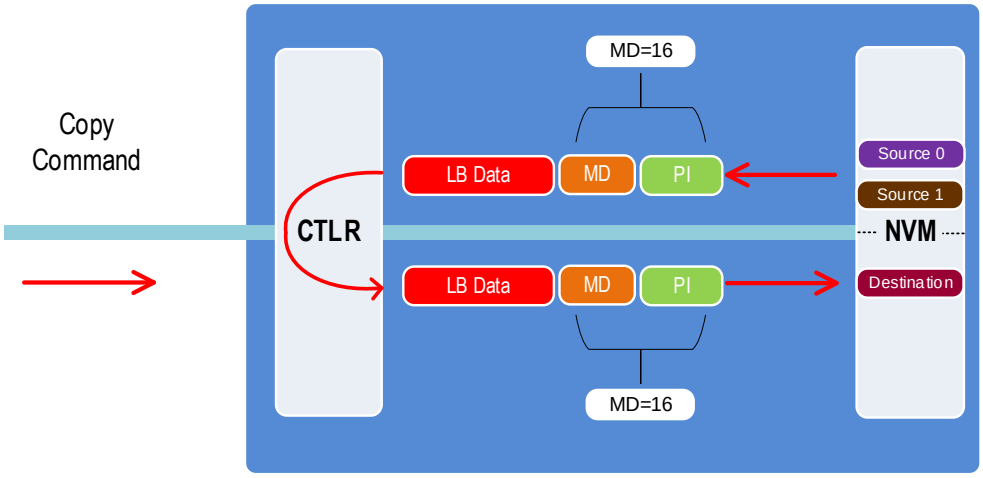
- The logical block data and metadata are transferred from the controller to the NVM.
- As the logical block data and metadata passes through the controller, the PI is handled as described in section 5.3.2.1.

Figure 169: PI Processing for Copy MD=8 Pass-through



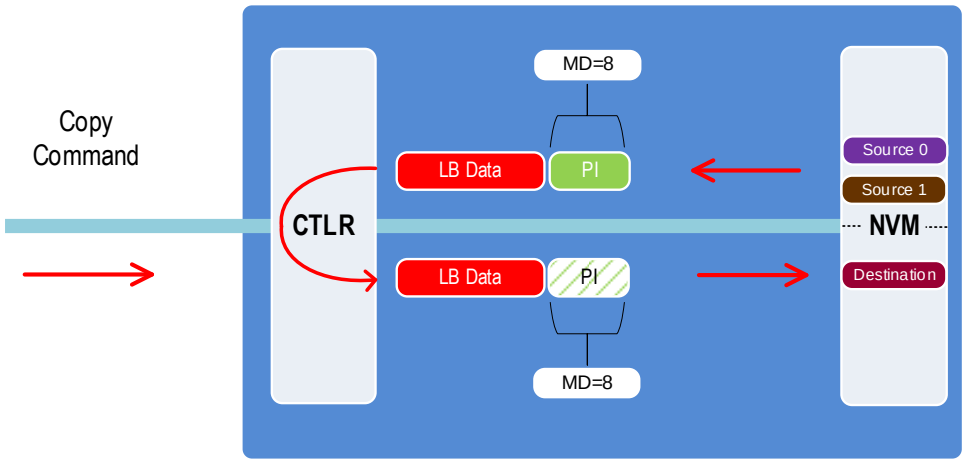
MD=8, PI, PRINFOR.PRACT=PRINFOW.PRACT=0 : PI pass-through

Figure 170: PI Processing for Copy MD=16 Pass-through



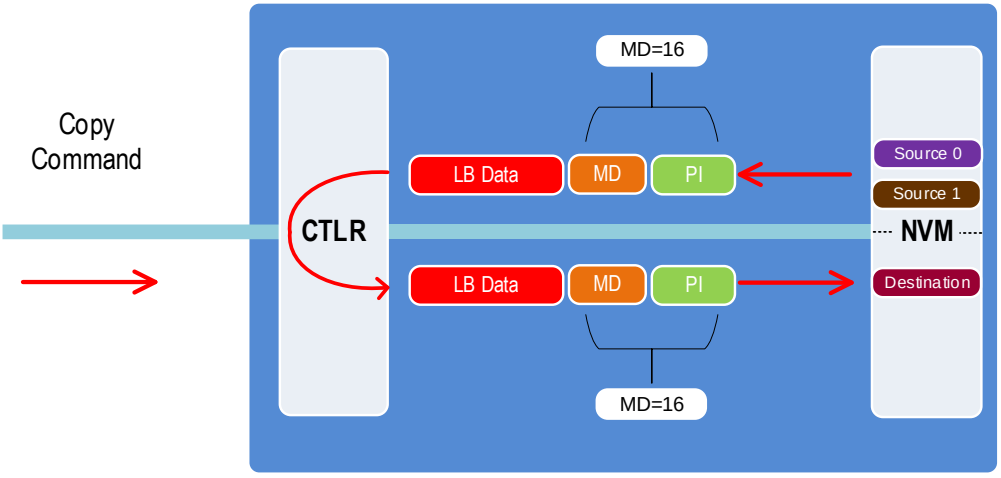
MD=16, PI, PRINFOR.PRACT=PRINFOW.PRACT=0 : PI pass-through

Figure 171: PI Processing for Copy MD=8 Replace



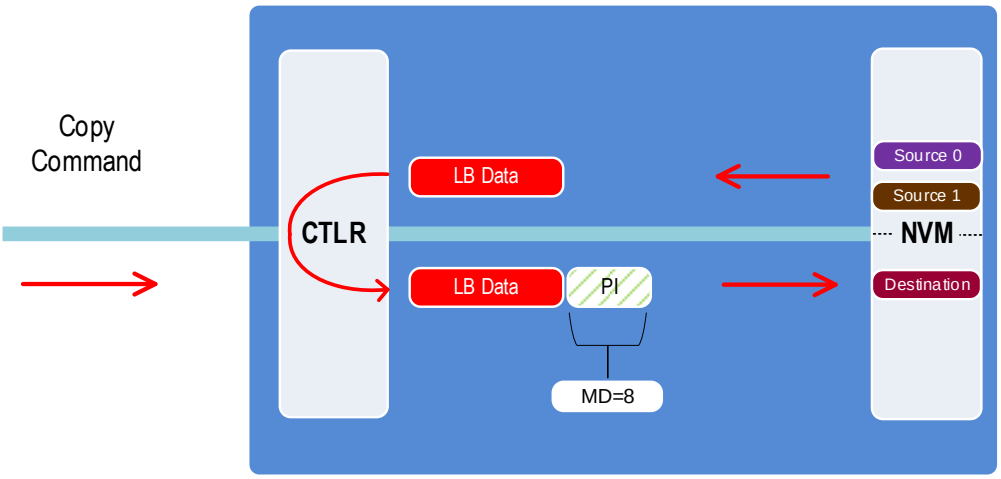
MD=8, PI, PRINFOR.PRACT=PRINFOW.PRACT=1 : PI replace

Figure 172: PI Processing for Copy MD=16 Replace



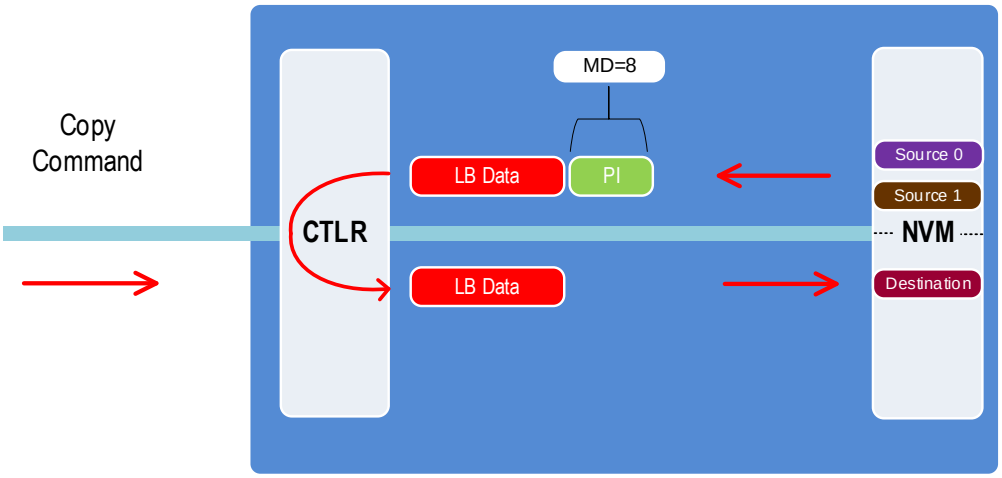
MD=16, PI, PRINFOR.PRACT=PRINFOW.PRACT=1 : PI replace

Figure 173: PI Processing for Copy MD=8 Insert



destination MD=8, PI, PRINFOW.PRACT=1: PI insert

Figure 174: PI Processing for Copy MD=8 Strip



source MD=8, PI, PRINFOR.PRACT=1: PI strip

Figure 169, Figure 170, Figure 171, Figure 172, Figure 173, and Figure 174 shows six examples of PI processing for the Copy command where the Copy command in each example copies from two source regions in two different source namespaces to a destination region in a third destination namespace.

While user data is passing through the controller the data should never be unprotected (e.g., Calculate the PI data associated with the write portion of the copy operation occurs before verification and removal of PI data associated with the read portion of the copy operation). If the Guard field is recalculated, then that Guard field should be compared to the original Guard field (i.e., the Guard field associated with the read portion of the copy operation).

5.3.2.5.1 Protection Information and copying within the same namespace

If a Copy command uses Source Range Entries Copy Descriptor Format 0h or 1h to request that user data in a Source Range be copied to the same namespace and that namespace is formatted with protection information (PI), then that user data is copied only if the PRINFOR.PRACT bit and the PRINFOW.PRACT bit in the Copy command have the same value (refer to section 3.3.2.3):

- If the PRINFOR.PRACT bit and the PRINFOW.PRACT bit are both set to '1', then as the logical block data and metadata pass through the controller, the PI is replaced with PI that is generated by the controller as shown for the examples in Figure 169 and Figure 170; and
- If the PRINFOR.PRACT bit and the PRINFOW.PRACT bit are both cleared to '0', then the logical block data and metadata pass through the controller without change to the PI as shown for the examples in Figure 171 and Figure 172.

If the PRINFOR.PRACT bit and the PRINFOW.PRACT bit do not have the same value, then the Copy command is aborted with a status code of Invalid Field in Command (refer to section 3.3.2.3).

PI checks for copied user data are performed as described in section 5.3.2.5.

5.3.2.5.2 Protection Information and copying across different namespaces

This section applies to Copy commands that use either Source Range Entries Copy Descriptor format 2h or 3h to copy data across multiple namespaces and/or within the same namespace. Refer to section 5.3.2.5.1 for Copy commands that use Source Range Entries Copy Descriptor Format 0h or 1h.

Controller processing of protection information (PI) as part of a copy operation depends on the format of the destination namespace, the format of each source namespace and the values of the PRINFOR.PRACT bit and the PRINFOW.PRACT bit in the Copy command.

If the destination namespace and a source namespace are both formatted with PI and the two namespaces have matching namespace formats for copy (refer to section 3.3.2.4.1), then:

- if the PRINFOR.PRACT bit and the PRINFOW.PRACT bit are both set to '1', then as the logical block data and metadata pass through the controller, the PI is overwritten with PI that is generated by the controller as shown for the examples in Figure 169 and Figure 170; and
- if the PRINFOR.PRACT bit and the PRINFOW.PRACT bit are both cleared to '0' then the logical block data and metadata pass through the controller without change as shown for the examples in Figure 171 and Figure 172.

All other cases in which the destination namespace and a source namespace are both formatted with PI result in the controller aborting the Copy command with an error status (refer to section 3.3.2.4.2).

If the destination namespace is formatted with PI, a source namespace is formatted without PI, and:

- the two namespaces have corresponding PI formats for copy (refer to section 3.3.2.4.1); and
- the PRINFOW.PRACT bit is set to '1',

then as the logical block data passes through the controller, the controller generates and appends PI to the end of the logical block data as shown for the example in Figure 173. All other cases in which the destination namespace is formatted with PI and a source namespace is formatted without PI result in the controller aborting the Copy command with an error status (refer to section 3.3.2.4.2), including cases where the metadata size of the destination namespace is larger than the PI size of that namespace.

If the destination namespace is formatted without PI, a source namespace is formatted with PI, and:

- the two namespaces have corresponding PI formats for copy; and
- the PRINFOR.PRACT bit is set to ‘1’,

then as the logical block data and metadata pass through the controller, the PI is stripped, and the controller only writes the logical block data to the NVM as shown for the example in Figure 174. All other cases in which the destination namespace is formatted without PI and a source namespace is formatted with PI result in the controller aborting the Copy command with an error status (refer to section 3.3.2.4.2), including cases where the metadata size of a source namespace is larger than the PI size of that namespace.

PI checks for copied user data are performed as described in section 5.3.2.5.

5.3.3 Control of Protection Information Checking - PRCHK

Checking of protection information consists of the following operations performed by the controller.

- If the Guard Check bit 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 for the protection information format (refer to section 5.3.1) computed over the logical block data.
- If the Application Tag Check bit 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 or the Expected Logical Block Application Tag Mask (ELBATM) field. If a Storage Tag field is defined in the protection information (refer to section 5.3.1.4) and the Storage Tag Check bit in the command is set to ‘1’, then the controller compares unmasked bits in the Storage Tag field to the Logical Block Storage Tag (LBST) field of the command. A bit in the Storage Tag field is masked (i.e., not compared) if the corresponding bit is cleared to ‘0’ in the Logical Block Storage Tag Mask (LBSTM) field in the NVM Command Set Identify Namespace data structure (refer to Figure 118). Implementations may limit support for the bits that are allowed to be masked in the Logical Block Storage Tag Mask as described in the Storage Tag Masking Level Attribute field (refer to Figure 118).
- If the Reference Tag is defined (refer to Figure 119) then:
    - If the Reference Tag Check bit of the PRCHK field is set to ‘1’ and the namespace is formatted for Type 1 or Type 2 protection, then the controller compares the Logical Block Reference Tag to the computed reference tag. The computed reference tag depends on the Protection Type:
        - If the namespace is formatted for Type 1 protection, the value of the computed reference tag for the first logical block 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, and the computed reference tag is incremented for each subsequent logical block. The controller shall complete the command with a status of Invalid Protection Information if the ILBRT field or the EILBRT field does not match the value of the least significant bits of the SLBA field sized to the number of bits in the Logical Block Reference Tag (refer to section 5.3.1.4).

          Note: 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 the host to initialize the ILBRT or EILBRT field to the

least significant bits of the LBA sized to the number of bits in the Logical Block Reference Tag when Type 1 protection is used.
    - If the namespace is formatted for Type 2 protection, the value of the computed reference tag for the first logical block 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, and the computed reference tag is incremented for each subsequent logical block.
- If the Reference Tag Check bit of the PRCHK field is set to '1' and the namespace is formatted for Type 3 protection, then the controller:
    - should not compare the protection Information Reference Tag field to the computed reference tag; and
    - may ignore the ILBRT and EILBRT fields. If a command is aborted as a result of the Reference Tag Check bit of the PRCHK field being set to '1', then that command shall be aborted with a status code of Invalid Protection Information.
- Incrementing a computed reference tag with all bits set to '1' produces a value with all bits cleared to '0' (i.e., the computed reference tag rolls over to 0h).

Protection checking may be disabled as a side effect of the value of the protection information Application Tag and Logical Block Reference Tag, if defined, 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 and Logical Block Reference Tag, if defined, have all bits set to '1'.

Inserted protection information consists of the computed CRC for the protection information format (refer to section 5.3.1) in the Guard field, the LBAT field value in the Application Tag field, the LBST field value in the Storage Tag field, if defined, and the computed reference tag in the Logical Block Reference Tag.

5.4 Key Per I/O

The Key Per I/O capability operates as defined by the NVM Express Base Specification with the exceptions specified by this section. The Trusted Computing Group Security Subsystem Key Per I/O specification makes use of this NVM Express capability.

Any I/O command specifying the key tag (i.e., CETYPE value of KPIOTAG) that does not have:

a) the SLBA field in that command aligned on a logical block boundary specified by the KPIODAAG field in the Identify Namespace data structure (refer to Figure 114); and
b) the range of LBAs accessed by that command specify a granularity specified by the KPIODAAG field,

then the controller shall abort that command with a status code of Invalid Field in Command.

5.5 LBA Format List Structure

To create or format a namespace, a host specifies an LBA Format (i.e., the Format Index) referencing the LBA Format list in the NVM Command Set Identify Namespace data structure. The LBA Format list has a structure as is illustrated in Figure 175. The NLBAF field, a 0-based number (i.e., at least one is required

to be defined), identifies the number of LBA Formats that have the same capabilities used to format and create a namespace (i.e., the green LBA Formats). The Identify command provides the ability to access these same capabilities in a namespace data structure by specifying an NSID of FFFFFFFFh.

The NULBAF field, a 1-based number (i.e., none may be defined), identifies the number of LBA Formats that have unique attributes (i.e., that may not be the same capabilities as other LBA Formats) used to format and create a namespace (i.e., the orange LBA Formats). A host should use the Identify command with a CNS value of 09h to access the capabilities of a specific LBA Format for the NVM Command Set Identify Namespace data structure. A host should use the Identify command with a CNS value of 0Ah to access the capabilities of a specific LBA Format for an I/O Command Set specific Identify Namespace data structure for the NVM Command Set.

Figure 176 shows the association of CNS values of the Identify command that provides the ability to access these same capabilities to specific LBA Format entries referenced by the NLBAF field and NULBF field.

The maximum number of LBA formats allowed to be supported is:

- 16 if the LBA Format Extension Enable (LBAFEE) field is cleared to 0h in the Host Behavior Support feature (refer to the Host Behavior Support section in the NVM Express Base Specification); or
- 64 if the LBAFEE field is set to 1h in the Host Behavior Support feature (refer to the Host Behavior Support section in the NVM Express Base Specification).

The total number of LBA formats supported is the sum of the values represented by the NLBAF field and the NULBAF field. A Format Index is valid if the value is less than the sum of the values represented by the NLBAF field and the NULBAF field. An LBA Format that is supported, but not currently available is indicated by clearing the LBA Data Size field to 0h for that LBA Format.

Figure 175: LBA Format List Structure

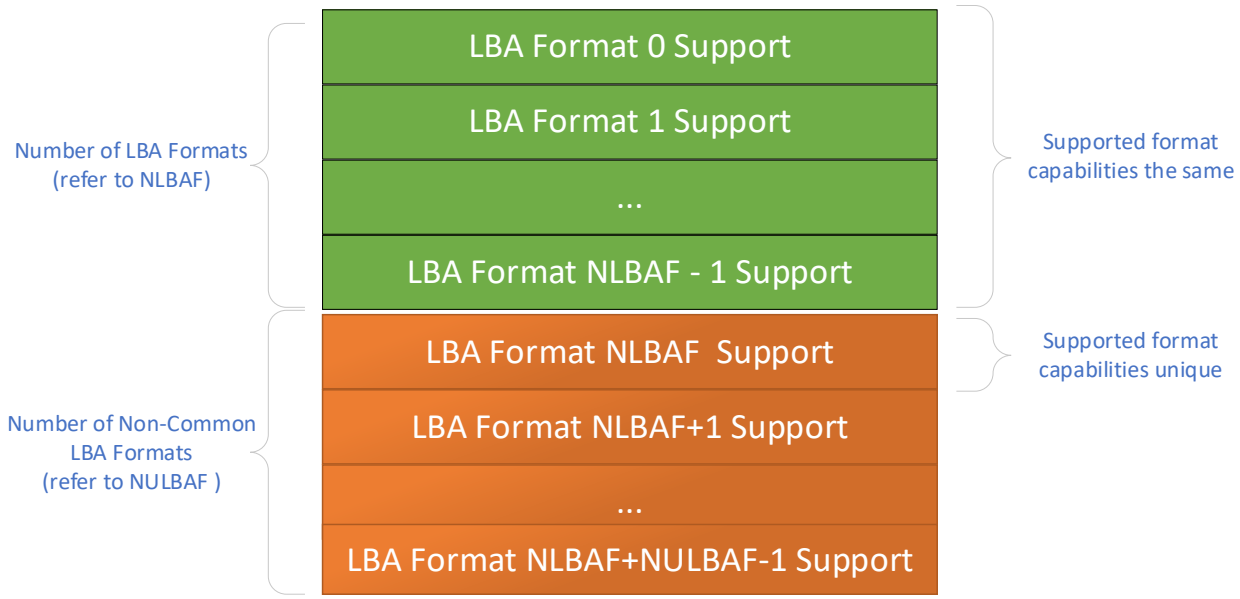


Figure 176: LBA Format List Entries Applicability to Identify Command CNS Value

CNS Value	Returned data is associated with LBA Format entries referenced by only the NLBAF field	Returned data is associated with LBA Format entries referenced by both the NLBAF field and NULBAF field
00h	Yes	No
05h	Yes	No
08h	Yes	No
09h	No	Yes
0Ah	No	Yes

5.6 LBA Migration Queue

As defined by the NVM Express Base Specification, a User Data Migration Queue allows a controller to post entries that identify user data modifications due to the processing of commands by the controller associated to the User Data Migration Queue. For the NVM Command Set:

- user data is logical blocks and posted entries indicate:
    - data in the logical blocks has changed (e.g., modified by a Write command);
    - data in the logical blocks may have changed (e.g., a Write command was processed that targeted the logical blocks but there was no change to the data in those logical blocks);
    - are deallocated (i.e., the logical block transitioned from being allocated to deallocated); or
    - may have been deallocated (i.e., the command requested to deallocate logical blocks that were already deallocated);

  and
- the User Data Migration Queue is referred to as the LBA Migration Queue.

If logging has been started in an LBA Migration Queue due to a Track Send command (refer to the NVM Express Base Specification), the first entry posted indicates that the logging has started. Subsequent entries are able to indicate if an entry is:

- the last entry due to the controller associated with the LBA Migration Queue being suspended as a result of a Migration Send command (refer to the NVM Express Base Specification);
- the last entry due to logging being stopped as a result of the controller processing a command that is defined to stop logging (e.g., the Track Send command (refer to the NVM Express Base Specification));
- the controller detecting that the LBA Migration Queue has become full; or
- the first entry due to the controller associated with the LBA Migration Queue being resumed while suspended as a result of a Migration Send command.

5.6.1 LBA Migration Queue Entries

The LBA Migration Queue Format (LBAMQF) field in the I/O Command Set specific Identify Controller data structure (refer to Figure 120) defines the supported format for entries in an LBA Migration Queue. Figure 177 defines the supported format.

A controller may aggregate the results of multiple commands processed by a controller into a single entry. For example:

- If the controller processes three sequential Write commands, then the controller may post a single entry that incorporates all of the logical blocks written.
- If the controller processes the sequential commands that modify the same LBA range, then the controller may post a single entry for the LBA range with the result from the last command of that set of sequential commands.

The controller shall only post an entry into an LBA Migration queue if the processing for the command reported by the entry has taken effect, which may be before the completion for that command is posted. For example, if a Write command is processed by a controller that results in the posting of an entry in the LBA Migration queue that has:

- LBAINR bit cleared to '0';
- the SLBA field set to the SLBA field of that Write command; and
- the NLB field set to the NLB field of that Write command,

then the host may issue a Read command to obtain the logical blocks written even if the completion for that Write command is not posted.

If:

- the controller processes a command that is a request to modify a logical block; and
- that modification results:
    - in the logical block remaining in the same condition (e.g., host deallocates a logical block that is already deallocated); or
    - the logical block has the exact same data (e.g., the controller processes a Write Zeroes command for a logical block that already is written with zero data),

then the controller may or may not report that logical block in an entry in the LBA Migration Queue.

Figure 177: LBA Migration Queue Entry Type 0

Bytes	Description
03:00	**Namespace Identifier (NSID):** This field indicates the namespace identifier associated with the logical blocks reported in this entry. If the LBACIR field is set to 10b, then this field shall be cleared to 0h and should be ignored by the host.
07:04	**Number of Logical Blocks (NLB):** This field indicates the number of logical blocks reported by this entry. This is a 0's based value. If the LBACIR field is set to 10b, then this field shall be cleared to 0h and should be ignored by the host.
15:08	**Starting LBA (SLBA):** This field indicates the 64-bit address of the first logical block of data reported by this entry. If the LBACIR field is non -zero, then this field shall be cleared to 0h and should be ignored by the host.
30:16	Reserved

Figure 177: LBA Migration Queue Entry Type 0

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>31</td><td>
LBA Migration Queue Attributes (LBAMQA): This field indicates attributes associates with the LBE Migration Queue entry.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>07:06</td><td>
LBA Change Information Attribute (LBACIR): This field indicates attributes associated with the reporting of the LBA range in this entry.

If the ESA field is cleared to 000b, then this field shall be cleared to 00b.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>This entry is reporting logical blocks that have changed in the reported namespace. The SLBA field and the NLB field identify the LBA range.</td></tr>
<tr><td>01b</td><td>This entry is reporting that all logical blocks in the reported namespace have changed.</td></tr>
<tr><td>10b</td><td>This value indicates that no LBA range is being reported by this entry.</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>05</td><td>
Deallocated LBAs (DLBA): If this bit is set to ‘1’, then: the logical blocks reported by this entry have been deallocated.

If this bit is cleared to ‘0’, then the logical blocks reported by this entry have been modified (e.g., been written or deallocated).
</td></tr>
<tr><td>04</td><td>Reserved</td></tr>
<tr><td>03:01</td><td>
Entry Sequence Attribute (ESA): This field specifies the attribute of this entry related to starting and stopping of the posting of entries into the LBA Migration Queue.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>
This entry is not the:
• first entry placed into the LBA Migration Queue since:
  ○ logging was started; or
  ○ the controller associated with the LBA Migration Queue resumed from a suspended state;
and
• last entry placed into the LBA Migration Queue since logging was stopped or the controller associated with the LBA Migration Queue was suspended.
</td></tr>
<tr><td>001b</td><td>
This is the first entry placed into the LBA Migration Queue since:
• logging was started; or
• the controller associated with the LBA Migration Queue resumed from a suspended state.
</td></tr>
<tr><td>010b</td><td>This is the last entry placed into the LBA Migration Queue as a result of logging being stopped due to a request from the host.</td></tr>
<tr><td>011b</td><td>This is the last entry placed into the LBA Migration Queue as a result of the controller associated with the LBA Migration Queue being suspended.</td></tr>
<tr><td>100b to 110b</td><td>Reserved</td></tr>
<tr><td>111b</td><td>This is the last entry placed into the LBA Migration Queue due to the LBA Migration Queue becoming full. This value notifies the host that the controller has stopped logging logical block changes into that LBA Migration Queue.</td></tr>
</table>
</td></tr>
<tr><td>00</td><td>Controller Data Queue Phase Tag (CDQP): This bit identifies whether a LBA Migration Queue entry is new as defined by the Controller Data Queue Phase Tag section in the NVM Express Base Specification.</td></tr>
</table>
</td></tr>
</table>

5.7 Namespace Management

Namespace Management capability operates as defined in the NVM Express Base Specification with additional capabilities specifically for the NVM Command Set.

The NVM Command Set supports reporting of Namespace Granularity as I/O Command Set specific Namespace Management capability content. The Namespace Granularity List defined in Figure 123 is requested by host software using the Identify command with CNS set to 16h.

If the controller supports reporting of Namespace Granularity, then the Namespace Granularity Descriptor List (refer to Figure 123) contains one or more Namespace Granularity Descriptors (refer to Figure 124) indicating the size granularity and the capacity granularity with which the controller creates namespaces.

The size granularity and the capacity granularity are hints which may be used by the host to minimize the capacity that is allocated for a namespace and that is not able to be addressed by logical block addresses. The granularities are used in specifying values for the Namespace Size (NSZE) field and Namespace Capacity (NCAP) field of the data structure used for the create operation of the Namespace Management command (refer to Figure 124).

If a Namespace Management command create operation specifies values such that:

a) the product of NSZE and the logical block size is an integral multiple of the Namespace Size Granularity;
b) the product of NCAP and the logical block size is an integral multiple of the Namespace Capacity Granularity; and
c) NSZE is equal to NCAP,

then the namespace is fully provisioned and all of the capacity allocated for the namespace is able to be addressed by logical block addresses, otherwise:

a) not all of the capacity allocated for the namespace is able to be addressed by logical block addresses; and
b) if the Namespace Management command is otherwise valid, then the controller shall not abort the command (i.e., the granularity values are hints).

5.8 NVM Command Set Media and Data Error Handling

Media and Data Error handling operates as described in the NVM Express Base Specification with the following extensions.

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.

5.9 Reservations

Reservations operate as defined in the NVM Express Base Specification with the additional I/O Command Set specific Command Behavior in the Presence of a Reservation defined in Figure 178.

Figure 178: 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
Read Command Group								
Compare Copy (source)[1] Read Verify	A	A	C	C	A	A	C	A
Write Command Group								
Copy (destination)[1] Dataset Management Write Write Uncorrectable Write Zeroes	C	C	C	C	C	A	C	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. Each source namespace of a Copy command is checked for reservation conflict as if accessed by a read command and the destination namespace of a Copy command is checked for reservation conflict as if accessed by a write command.

5.10 Sanitize Operations

Sanitize operates as defined in the NVM Express Base Specification. NVM Command Set specific definitions and extensions are defined in this section.

Following a successful sanitize operation, the values of user data, protection information, and non-PI metadata that result from an audit (refer to the NVM Express Base Specification) of the sanitization target are specified in Figure 179. If the controller deallocates logical blocks after successful completion of a sanitize operation, then values read from deallocated logical blocks are described in section 3.3.3.2.1. 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 179: Sanitize Operation Types – User Data Values

Sanitize Operation Type	User Data
Block Erase	Vendor specific value
Crypto Erase	Indeterminate value

Figure 179: Sanitize Operation Types – User Data Values

Sanitize Operation Type	User Data
Overwrite	Refer to Sanitize Operations – Overwrite Mechanism in the NVM Express Base Specification

5.10.1 Media Verification

While the sanitization target is in the Media Verification state (refer to the Sanitize Operation State Machine Section in the NVM Express Base Specification), the controller processes Read commands as described in this section and shall not abort those commands with a status code of Sanitize In Progress.

If:

- the controller processes a Read command that does not specify any Protection Information (PI) checking (i.e., the PRCHK field is cleared to 000b; refer to Figure 11); and
- for each LBA specified by that command for which media is allocated, the controller is able to read data from the media,

then:

- for each LBA specified by that command for which media is allocated, the controller:
    - shall ignore data integrity errors, if any (e.g., shall not abort that command with a status code of Unrecovered Read Error if the controller is able to read that media);
    - shall return data that is read from that media; and
    - may return different data for successive reads (i.e., without any writes between those reads) of the same LBA (e.g., to obscure media reliability);
- for each LBA specified by that command for which media is not allocated, the controller shall return data or abort that command with a status code of Deallocated or Unwritten Logical Block as described in section 3.3.3.2.1; and
- the controller shall complete that command with a status code of Successful Media Verification Read if the command has not been aborted with a different status code (e.g., Deallocated or Unwritten Logical Block).

If the controller processes a Read command that does not specify any PI checking and is unable to read the data from the media for any LBA specified by that command for which media is allocated, then the controller shall abort the command with a status code of Unrecovered Read Error.

If the controller processes a Read command specifying PI checking (i.e., the PRCHK field is set to a non-zero value), then the controller shall abort that command with a status code of Invalid Field in Command.

5.11 Streams

Streams operate as defined in the NVM Express Base Specification. The unit of granularity for the NVM Command Set specific definition of the Stream Write Size (SWS) field is in logical blocks.