NVM Express®

NVM Command Set Specification

Revision 1.0b

January 6th, 2022

Please send comments to info@nvmexpress.org

NVM Express® NVM Command Set Specification, revision 1.0b is available for download at http://nvmexpress.org. http://nvmexpress.org. The NVM Express NVM Command Set Specification, revision 1.0a incorporates NVM Express NVM Command Set Specification, revision 1.0 (refer to https://nvmexpress.org/changes-in-nvm-express-revision-2-0 for details), ECN 001, and ECN102.

SPECIFICATION DISCLAIMER

LEGAL NOTICE:

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

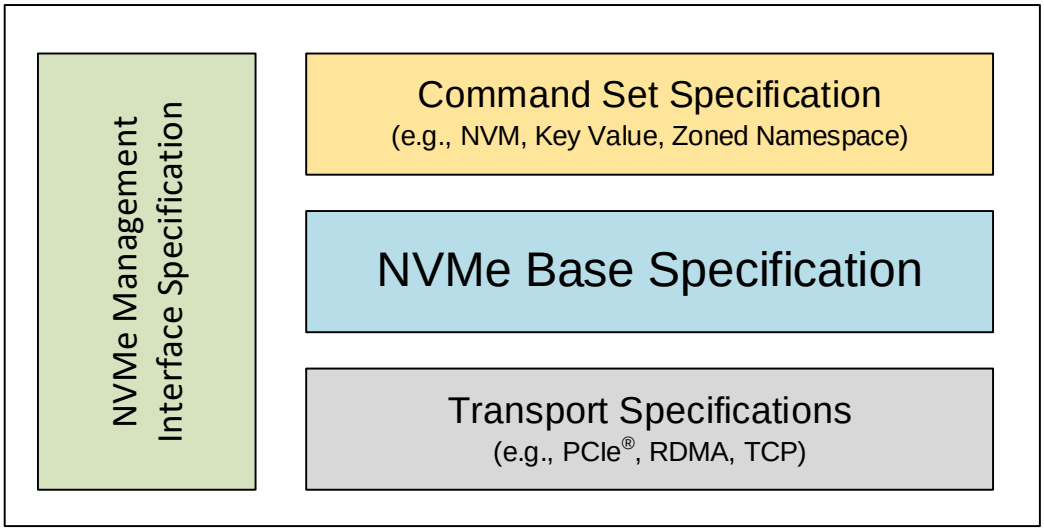
NVM Express® (NVMe®) Base specification defines an interface for host software to communicate with non-volatile memory subsystems over a variety of memory-based transports and message-based transports.

This document defines a specific NVMe I/O Command Set, the NVM Command Set, which extends the NVMe 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 NVMe 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 NVMe 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. NVMe Base Specification
3. NVMe transport specifications
4. NVMe I/O command set specifications
5. NVMe-MI specification

1.3 Conventions

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

1.4 Definitions

1.4.1 Definitions from the NVMe Base Specification

This specification uses the definitions in the NVMe Base Specification.

1.4.2 Definitions in the NVMe 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 Format data structure and the Extended LBA Format data structure.

1.4.2.3 SMART Data Units Read Command

The Compare command, Read command, and Verify command.

1.4.2.4 SMART Host Read Command

The Compare command, Copy command, and Read command.

1.4.2.5 User Data Format

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

1.4.2.6 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 148.

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 2.0. Available from http://www.nvmexpress.org.

2 NVM Command Set Model

The NVMe Base Specification defines a property level interface for host software to communicate with a non-volatile memory 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 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 a collection of NVM and is as defined in the NVMe Base Specification.

The Identify Namespace data structure (refer to Figure 97) contains related fields reporting the Namespace Size, Capacity and Utilization:

- The Namespace Size (NSZE) field defines the total size of the namespace in logical blocks (LBA 0 through 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.

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

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

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

- shall report a value in the Namespace Capacity field that is equal to the Namespace Size; 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 it is written with:

- a Write command;
- a Write Uncorrectable command; or
- a Write Zeroes command that does not deallocate the logical block (refer to section 3.2.3.2.1).

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

- a Write command not addressing the logical block; or
- a Write Zeroes command not addressing the logical block (refer to section 3.2.3.2.1).

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

- a Dataset Management command (refer to section 3.2.3); or
- a Write Zeroes command addressing the logical block (refer to section 3.2.3.2.1); or
- a sanitize operation.

If the controller supports Asymmetric Namespace Access Reporting (i.e., bit 3 set to '1' in the CMIC field in the Identify Controller data structure (refer to the NVMe Base Specification)), then the NUSE field (refer to Figure 97) and the NVMCAP field (refer to Figure 97) 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 NVMe Base Specification). The Namespace Attribute Changed event is not generated for changes to these fields that result from ANA state changes as described in the Asynchronous Event Request command section in the NVMe 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 AWUN (refer to section 2.1.4.1), 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 is submitted for LBA *x* and there is a Write also submitted for LBA *x*, there is no guarantee of the order of completion for those commands (the Read may finish first or the Write may finish first). If there are ordering requirements between these commands, host software or the associated application is required to enforce that ordering above the level of the controller.

2.1.3 Fused Operation

Fused operations are defined in the NVMe 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 is successful, then the LBA range is updated with the data provided in the Write command. If the Compare operation is not successful, then the Write operation is aborted with a status of Command Aborted due to Failed Fused Command and the contents in the LBA range are not modified. If the Write operation is not successful, the Compare operation completion status is unaffected.

The LBA range, if used, shall be the same for the two commands. If the LBA ranges do not match, the commands should be aborted with status of Invalid Field in Command;

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 2.1.4.3). Controllers may abort a Compare and Write fused operation that is larger than ACWU/NACWU or that crosses an Atomic Boundary with an error of Atomic Write Unit Exceeded.

2.1.4 Atomic Operation

Figure 4 is an overview of the parameters that define the controller's support for atomic operations. These parameters may affect command behavior and execution order based on write size (on a per controller or a per namespace basis).

Figure 4: Atomicity Parameters

	Parameter Name	Value[1]
Controller Atomic Parameters (refer to Figure 99)	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 97)	Namespace Atomic Write Unit Normal (NAWUN)	≥ AWUN
	Namespace Atomic Write Unit Power Fail (NAWUPF)	≥ AWUPF ≤ NAWUN
	Namespace Atomic Compare and Write Unit (NACWU)	≥ ACWU
Namespace Atomic Boundary Parameters (refer to the Identify Namespace data structure in Figure 97)	Namespace Atomic Boundary Size Normal (NABSN)	≥ NAWUN
	Namespace Atomic Boundary Offset (NABO)	≤ NABSN ≤ NABSPF
	Namespace Atomic Boundary Size Power Fail (NABSPF)	≥ NAWUPF

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

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

An NVM subsystem may report per namespace values for these fields that are specific to the namespace format and are indicated in the Identify Namespace data structure (refer to Figure 97). 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 99).

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

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

2.1.4.1 AWUN/NAWUN

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

If a write command is submitted with size less than or equal to the AWUN/NAWUN value and the write command does not cross an atomic boundary (refer to section 2.1.4.3), then the host is guaranteed that the write command is atomic to the NVM with respect to other read or write commands. If a write command is submitted with size greater than the AWUN/NAWUN value or crosses an atomic boundary, then there is no guarantee of command atomicity. AWUN/NAWUN does not have any applicability to write errors caused by power failure or other error conditions (refer to section 2.1.4.2).

The host may indicate that AWUN and NAWUN are not necessary by configuring the Write Atomicity Normal feature (refer to section 4.1.3.4), which may result in higher performance in some implementations.

2.1.4.1.1 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.

2.1.4.2 AWUPF/NAWUPF

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

If a write command is submitted with size less than or equal to the AWUPF/NAWUPF value and the write command does not cross an atomic boundary (refer to section 2.1.4.3), the controller guarantees that if the

command fails due to a power failure or other error condition, then subsequent read commands for the logical blocks associated with the write command shall return one of the following:

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

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

2.1.4.2.1 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.

2.1.4.2.2 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 NVMe Base Specification,

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

2.1.4.3 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).

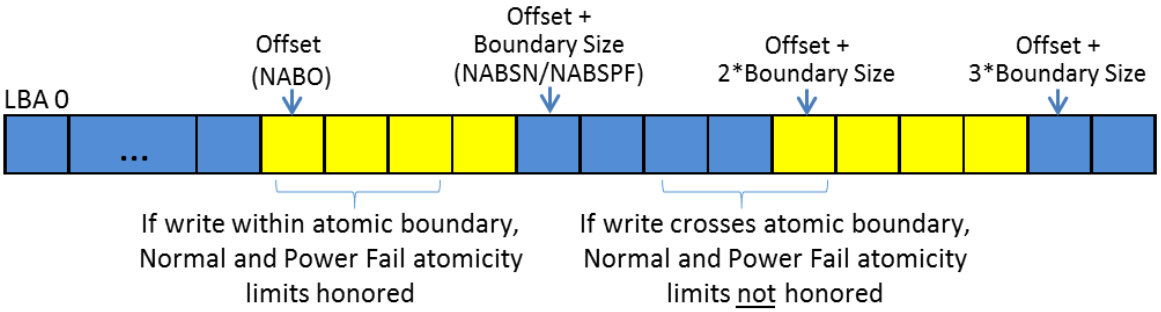
The boundary size shall be greater than or equal to the corresponding atomic write size (i.e., NABSN/NABSPF shall be greater than or equal to NAWUN/NAWUPF, respectively). NABO shall be less than or equal to NABSN and NABSPF.

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

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

Figure 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



2.1.5 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 9 and Figure 10.

Figure 9: 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 5.2.
<table>
<tr><th rowspan="2">PRACT Value</th><th colspan="2">Metadata Size</th><th rowspan="2">Description</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>
<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 5.2.
<table>
<tr><th>Bit</th><th>Definition</th></tr>
<tr><td>02</td><td>Guard Check: If set to ‘1’ enables protection information checking of the Guard field. If cleared to ‘0’, the Guard field is not checked.</td></tr>
<tr><td>01</td><td>Application Tag Check: If set to ‘1’ enables protection information checking of the Application Tag field. If cleared to ‘0’, the Application Tag field is not checked.</td></tr>
<tr><td>00</td><td>Reference Tag Check: If set to ‘1’ enables protection information checking of the Logical Block Reference Tag field. If cleared to ‘0’, the Logical Block Reference Tag field is not checked.</td></tr>
</table></td></tr>
</table>

Figure 10: Storage Tag Check Definition

Bits	Description
00	**Storage Tag Check:** This bit specifies the checking requirements for the Storage Tag field, if defined. If set to ‘1’, then protection information checking of the Storage Tag field is enabled. If cleared to ‘0’, the Storage Tag field is not checked. Refer to section 5.2. If the Storage Tag Size (STS) field is cleared to 0h (refer to Figure 101), 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 5.8.3.

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 within the command.

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

If the namespace is formatted to use end-to-end data protection (refer to section 5.2), then the first bytes or last bytes of the metadata is used for protection information (specified as part of the Format NVM command).

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 NVMe Base Specification. Additionally, Figure 11 and Figure 12 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 11: I/O Controller – Admin Command Support

Command	Command Support Requirements [1]
Get LBA Status	O
Notes: 1. O = Optional, M = Mandatory, P = Prohibited	

Figure 12: I/O Controller – NVM Command Set Support

Command	Command Support Requirements [1]
Write	M
Read	M
Write Uncorrectable	O
Compare	O
Copy	O
Write Zeroes	O
Verify	O
Vendor Specific	O
Notes: 1. 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 NVMe Base Specification. Additionally, Figure 13 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 13: I/O Controller – NVM Log Page Support

Log Page Name	Log Page Support Requirements [1]
LBA Status Information	O
Notes: 1. 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 NVMe Base Specification. Additionally, Figure 13 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 14: I/O Controller – Feature Support

Feature Name	Feature Support Requirements [1]	Logged in Persistent Event Log
LBA Range Information	O	NR
Error Recovery	M	O
Write Atomicity Normal	M	O
LBA Status Information Attributes	O	O

Notes:
1. 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 – Command Format section in the NVMe 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 NVMe Base Specification. The command specific fields in the SQE and CQE 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 Queueing Data Structures section of the NVMe Base Specification.

3.1.1 Common Command Format

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

3.1.2 NVM Command Set Specific Status Values

Figure 15: Status Code – Generic Command Status Values

Value	Description
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 to revision 1.2.1 of the NVMe Base Specification or earlier may use the status code value of 15h to indicate SGL Data Block Granularity Invalid.
80h	**LBA Out of Range:** The command references an LBA that exceeds the size of the namespace.

Figure 16: Status Code – Command Specific Status Values

Value	Description	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
84h to BFh	Reserved	

Figure 17: Status Code – Media and Data Integrity Error Values

Value	Description
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 NVM Command Set Commands

The NVM Command Set includes the commands listed in Figure 18. This section describes the definition for each of the commands defined by this specification. Commands are submitted as described in the NVMe Base Specification. Physical region page (PRP) entries (refer to the Data Layout section of the NVMe Base Specification) and scatter gather lists (SGL) (refer to the Data Layout section of the NVMe 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 4.1.3.3. If the host does not specify a time limit should be applied, then the controller should apply all error recovery means to complete the operation.

Figure 18: Opcodes for NVM Commands

Opcode by Field			Combined Opcode [1]	Command [2]	Reference Section
(07)	(06:02)	(01:00)			
Standard Command	Function	Data Transfer [3]			
0b	000 00b	00b	00h	Flush [4]	Base
0b	000 00b	01b	01h	Write	3.2.6
0b	000 00b	10b	02h	Read	3.2.4
0b	000 01b	00b	04h	Write Uncorrectable	3.2.7
0b	000 01b	01b	05h	Compare	3.2.1
0b	000 10b	00b	08h	Write Zeroes	3.2.8
0b	000 10b	01b	09h	Dataset Management	3.2.3
0b	000 11b	00b	0Ch	Verify	3.2.5
0b	000 11b	01b	0Dh	Reservation Register	Base
0b	000 11b	10b	0Eh	Reservation Report	Base
0b	001 00b	01b	11h	Reservation Acquire	Base
0b	001 01b	01b	15h	Reservation Release	Base
0b	001 10b	01b	19h	Copy	3.2.2
Vendor Specific					
1b	n/a	NOTE 3	80h to FFh	Vendor specific	

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.
5. Key: Base = NVMe Base Specification

3.2.1 Compare command

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

If metadata is provided, then a comparison is also performed for the metadata, excluding protection information. The command may specify protection information to be checked as described in section 5.2.2.4.

The command uses Command Dword 2, Command Dword 3, Command Dword 10, Command Dword 11, Command Dword 12, Command Dword 14, and Command Dword 15 fields. If the command uses PRPs for the data transfer, then the Metadata Pointer, PRP Entry 1, and PRP Entry 2 fields are used. If the command uses SGLs for the data transfer, then the Metadata SGL Segment Pointer and SGL Entry 1 fields are used. All other command specific fields are reserved.

Figure 19: 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 NVMe Base Specification for the definition of this field.

Figure 20: 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 NVMe Base Specification for the definition of this field.

Figure 21: Compare – Command Dword 2 and Dword 3

Bits	Description
63:48	Reserved
47:00	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 5.2.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored.

Figure 22: 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 23: Compare – Command Dword 12

Bits	Description
31	**Limited Retry (LR):** If set to '1', the controller should apply limited retry efforts. If cleared to '0', the controller should apply all available error recovery means to retrieve the data for comparison.
30	**Force Unit Access (FUA):** If 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 media; and 2) read the data and metadata, if any, from non-volatile media. If cleared to '0', then this bit has no effect.
29:26	**Protection Information Field (PRINFO):** Specifies the protection information action and check field, as defined in Figure 9. The Protection Information Action (PRACT) bit shall be cleared to '0'. If the Protection Information Check (PRCHK) field is non-zero, a check is performed on the logical block read from NVM (refer to section 5.2.2.4).
25	Reserved

Figure 23: Compare – Command Dword 12

Bits	Description
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 10.
23:16	Reserved
15:00	**Number of Logical Blocks (NLB):** This field specifies the number of logical blocks to be compared. This is a 0's based value.

Figure 24: Compare – Command Dword 14

Bits	Description
31:00	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 5.2.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored.

Figure 25: 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 5.2.
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 5.2.

3.2.1.1 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 26.

Figure 26: Compare – Command Specific Status Values

Value	Description
81h	**Invalid Protection Information:** The Protection Information Field (PRINFO) (refer to Figure 23) settings specified in the command are invalid for the Protection Information with which the namespace was formatted (refer to the PI field in Figure 78 and the DPS field in Figure 97) or the EILBRT field is invalid (refer to section 5.2.2.5).

3.2.2 Copy command

The Copy command is used by the host to copy data from one or more source logical block ranges to a single consecutive 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 27: 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 NVMe Base Specification for the definition of this field.

Figure 28: Copy – Command Dword 2 and Dword 3

Bits	Description
63:48	Reserved
47:00	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 5.2.1.4.1, to be used for the write portion of the copy operation. If the namespace is not formatted to use end-to-end protection information, then this field is ignored.

Figure 29: 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 30: Copy – Command Dword 12

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31</td><td>Limited Retry (LR): If set to ‘1’, the controller should apply limited retry efforts for the write portion of the copy operation. If cleared to ‘0’, 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 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 media before indicating command completion.

There is no implied ordering with other commands. If cleared to ‘0’, then this bit has no effect.</td></tr>
<tr><td>29:26</td><td>Protection Information Field Write (PRINFOW): Specifies the protection information action and check field, as defined in Figure 9, to be used for the write portion of the copy operation.</td></tr>
<tr><td>25</td><td>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 10, 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 NVMe Base Specification) used for the write portion of the copy operation.</td></tr>
<tr><td>19:16</td><td>Reserved</td></tr>
<tr><td>15:12</td><td>Protection Information Field Read (PRINFOR): Specifies the protection information action and check field, as defined in Figure 9, 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: Specifies the type of the Copy Descriptor Format that is used. The Copy Descriptor Format specifies the starting LBA, number of logical blocks, and parameters associated with the read portion of the operation.

<table>
<tr><th>Copy Descriptor Format Type</th><th>Description</th></tr>
<tr><td>0h</td><td>Source Range Entries Descriptor Format 0h is used (refer to Figure 34).</td></tr>
<tr><td>1h</td><td>Source Range Entries Descriptor Format 1h is used (refer to Figure 35).</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 31: 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 NVMe Base Specification).
15:00	Reserved

Figure 32: Copy – Command Dword 14

Bits	Description
31:00	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 5.2.1.4.1, to be used for the write portion of the copy operation. If the namespace is not formatted to use end-to-end protection information, then this field is ignored.

Figure 33: 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 namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.2.
15:00	**Logical Block Application Tag (LBAT):** This field specifies the Application Tag value for the write portion of the copy operation. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.2.

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

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 SDLBA. The Copy Descriptor Format type of the Source Range Entries is specified in the Descriptor Format field in Command Dword 12. 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 namespace specified by NSID is formatted to use 16b Guard Protection Information; and
c) the Descriptor Format is not cleared to 0h,

then the command shall be aborted with the 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 namespace specified by NSID is formatted to use 32b Guard Protection Information or 64b Guard Protection Information; and
c) the Descriptor Format is not set to 1h,

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

Figure 34 shows the Copy Descriptor Format 0h descriptor and an example with 128 Source Range entries.

Figure 34: Copy – Source Range Entries Descriptor Format 0h

<table>
<tr><th>Range</th><th>Bytes</th><th>Description</th></tr>
<tr><td rowspan="6">Source Range 0</td><td>07:00</td><td>Reserved</td></tr>
<tr><td>15:08</td><td>Starting LBA</td></tr>
<tr><td>19:16</td><td>Read Parameters as follows:
<table><tr><th>Bits</th><th>Description</th></tr><tr><td>31:16</td><td>Reserved</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>23:20</td><td>Reserved</td></tr>
<tr><td>27:24</td><td>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 5.2.1.4.1, to be used for the read portion of the copy operation for the LBAs specified in this Source Range entry. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.2.</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 namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.2.</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 namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.2.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td rowspan="8">Source Range 1</td><td>39:32</td><td>Reserved</td></tr>
<tr><td>47:40</td><td>Starting LBA</td></tr>
<tr><td>51:48</td><td>Read Parameters</td></tr>
<tr><td>55:52</td><td>Reserved</td></tr>
<tr><td>59:56</td><td>The variable sized ELBST and EILBRT fields</td></tr>
<tr><td>61:60</td><td>ELBAT</td></tr>
<tr><td>63:62</td><td>ELBATM</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td rowspan="8">Source Range 127</td><td>4071:4064</td><td>Reserved</td></tr>
<tr><td>4079:4072</td><td>Starting LBA</td></tr>
<tr><td>4083:4080</td><td>Read Parameters</td></tr>
<tr><td>4087:4084</td><td>Reserved</td></tr>
<tr><td>4091:4088</td><td>The variable sized ELBST and EILBRT fields</td></tr>
<tr><td>4093:4092</td><td>ELBAT</td></tr>
<tr><td>4095:4094</td><td>ELBATM</td></tr>
</table>

Figure 35 shows the Copy Descriptor Format 1h descriptor and an example with 102 Source Range entries.

Figure 35: Copy – Source Range Entries Descriptor Format 1h

<table>
<tr><th>Range</th><th>Bytes</th><th>Description</th></tr>
<tr><td rowspan="3">Source Range 0</td><td>07:00</td><td>Reserved</td></tr>
<tr><td>15:08</td><td>Starting LBA</td></tr>
<tr><td>19:16</td><td>Read Parameters as follows:
<table><tr><th>Bits</th><th>Description</th></tr><tr><td>31:16</td><td>Reserved</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>
</table>

Figure 35: Copy – Source Range Entries Descriptor Format 1h

Range	Bytes	Description
	25:20	Reserved
	35:26	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 5.2.1.4.1, to be used for the read portion of the copy operation. If the namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.2.
	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 namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.2.
	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 namespace is not formatted to use end-to-end protection information, then this field is ignored by the controller. Refer to section 5.2.
Source Range 1	47:40	Reserved
	55:48	Starting LBA
	59:56	Read Parameters
	65:60	Reserved
	75:66	The variable sized ELBST and EILBRT
	77:76	ELBAT
	79:78	ELBATM
	…	
Source Range 101	4047:4040	Reserved
	4055:4048	Starting LBA
	4059:4056	Read Parameters
	4065:4060	Reserved
	4075:4066	The variable sized ELBST and EILBRT
	4077:4076	ELBAT
	4079:4078	ELBATM
	4095:4080	Reserved

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 97), then the Copy command shall be aborted with a status code of Command Size Limit Exceeded.

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 97), then the Copy command shall be aborted with a status code of Command Size Limit Exceeded.

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 97), then the Copy command shall be aborted with a status code of Command Size Limit Exceeded.

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.

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 entry 0 are copied to the lowest numbered LBAs specified by the SDLBA field, the LBAs specified by Source Range entry 1 are copied to the next consecutively numbered LBAs specified by the

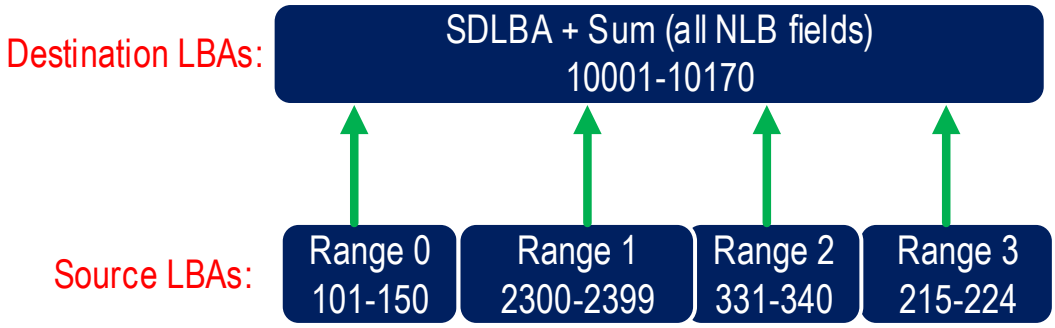
SDLBA field). The read operations and write operations used to perform the copy may operate sequentially or in parallel.

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 3.2.3.2.1.

If the namespace is formatted to use end-to-end protection information, then the protection information is handled as described in section 5.2.2.5.

Figure 36 shows an example of the relationship between the source LBAs and the destination LBAs.

Figure 36: Source LBA and Destination LBA Relationship Example



3.2.2.1 Command Completion

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

If the command completes with failure, then 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). 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 37.

Figure 37: Copy – Command Specific Status Values

Value	Description
81h	**Invalid Protection Information:** The protection information specified by the command is invalid due to: • The Protection Information Field Read (PRINFOR) field or Protection Information Field Write (PRINFOW) field (refer to Figure 30) 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 NVMe Base Specification and the DPS field in Figure 97) • the ILBRT field being invalid (refer to section 5.2.2.5); or • the EILBRT field in a Source Range Entry being invalid (refer to section 5.2.2.5)

Figure 37: Copy – Command Specific Status Values

Value	Description
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 NVMe Base Specification).
83h	**Command Size Limit Size 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.

3.2.3 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 38: 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 NVMe Base Specification for the definition of this field.

Figure 39: 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 40: Dataset Management – Command Dword 11

Bit	Description
31:03	Reserved
02	**Attribute – Deallocate (AD):** If set to '1', then the NVM subsystem may deallocate all provided ranges. The data returned for logical blocks that were deallocated is specified in section 3.2.3.2.1. 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 set to '1', then the dataset should be optimized for write access as an integral unit. The host expects to perform operations on all ranges provided as an integral unit for writes, indicating that if a portion of the dataset is written it is expected that all of the ranges in the dataset are going to be written.
00	**Attribute – Integral Dataset for Read (IDR):** If set to '1', then the dataset should be optimized for read access as an integral unit. The host expects to perform operations on all ranges provided as an integral unit for reads, indicating that if a portion of the dataset is read it is expected that all of the ranges in the dataset are going to be read.

If the Dataset Management command is supported, all combinations of attributes specified in Figure 40 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 41. The maximum case of 256 ranges is shown.

Figure 41: Dataset Management – Range Definition

Range	Bytes	Field
Range 0	03:00	Context Attributes
	07:04	Length in logical blocks
	15:08	Starting LBA
Range 1	19:16	Context Attributes
	23:20	Length in logical blocks
	31:24	Starting LBA
…		
Range 255	4083:4080	Context Attributes
	4087:4084	Length in logical blocks
	4095:4088	Starting LBA

3.2.3.1 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 NVMe 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 41) 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 41) 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 bit 2 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 bit 2 is cleared to '0' in the ONCS field, 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 bit 2 is set to '1' in the ONCS field, 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 bit 2 is cleared to '0' in the ONCS field, 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).

3.2.3.1.1 Dataset Management Processing Limits Example

For example, under the assumptions that the bit 2 in the ONCS field is set to '1' 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.

3.2.3.2 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 42: Dataset Management – Context Attributes

Attribute	Bits	Description
Command Access Size	31:24	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.
Reserved	23:11	Reserved
WP: Write Prepare	10	If set to '1', then the provided range is expected to be written in the near future.
SW: Sequential Write Range	09	If set to '1', then the dataset should be optimized for sequential write access. The host expects to perform operations on the dataset as a single object for writes.
SR: Sequential Read Range	08	If set to '1', then the dataset should be optimized for sequential read access. The host expects to perform operations on the dataset as a single object for reads.
Reserved	07:06	Reserved
AL: Access Latency	05:04	**Value** / **Definition** 00b – None. No latency information provided. 01b – Idle. Longer latency acceptable. 10b – Normal. Typical latency. 11b – Low. Smallest possible latency.
AF: Access Frequency	03:00	**Value** / **Definition** 0h – No frequency information provided. 1h – Typical number of reads and writes expected for this LBA range. 2h – Infrequent writes and infrequent reads to the LBA range indicated. 3h – Infrequent writes and frequent reads to the LBA range indicated. 4h – Frequent writes and infrequent reads to the LBA range indicated. 5h – Frequent writes and frequent reads to the LBA range indicated. 6h to Fh – Reserved

3.2.3.2.1 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 bits 2:0 in the DLFEAT field are set to 001b;
- all bytes set to FFh if bits 2:0 in the DLFEAT field are set to 010b; or

- either all bytes cleared to 0h or all bytes set to FFh if bits 2:0 in the DLFEAT field are set 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 Application Tag field, Storage Tag field, if defined, and the Logical Block Reference Tag value set to FFh (indicating the protection information shall not be checked).

Using the Error Recovery feature (refer to section 4.1.3.3), 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-4 and SCSI UNMAP command described in SBC-3.

3.2.3.3 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 are defined in Figure 43.

Figure 43: Dataset Management – Command Specific Status Values

Value	Description
20h	**Namespace is Write Protected:** The command is prohibited while the namespace is write protected (refer to the Namespace Write Protection section of the NVMe Base Specification).
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 NVMe 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 3.2.3.1). The controller shall not return this status value if bit 2 is set to '1' in the Optional NVM Command Support field in the Identify Controller data structure.

3.2.4 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 44: 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 NVMe Base Specification for the definition of this field.

Figure 45: 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 NVMe Base Specification for the definition of this field.

Figure 46: Read – Command Dword 2 and Dword 3

Bits	Description
63:48	Reserved
47:00	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 5.2.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored.

Figure 47: 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 48: Read – Command Dword 12

Bits	Description
31	**Limited Retry (LR):** If set to '1', the controller should apply limited retry efforts. If cleared to '0', the controller should apply all available error recovery means to return the data to the host.
30	**Force Unit Access (FUA):** If 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 media; and 2) return the data, and metadata, if any, that are read from non-volatile media. There is no implied ordering with other commands. If cleared to '0', then this bit has no effect.
29:26	**Protection Information Field (PRINFO):** Specifies the protection information action and check field, as defined in Figure 9.
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 10.
23:16	Reserved
15:00	**Number of Logical Blocks (NLB):** This field indicates the number of logical blocks to be read. This is a 0's based value.

Figure 49: Read – Command Dword 13

Bits	Description
31:08	Reserved

Figure 49: Read – Command Dword 13

<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 read.
<table>
<tr><th>Bits</th><th>Attribute</th><th>Definition</th></tr>
<tr><td>07</td><td>Incompressible</td><td>If set to ‘1’, then data is not compressible for the logical blocks indicated. If cleared to ‘0’, then no information on compression is provided.</td></tr>
<tr><td>06</td><td>Sequential Request</td><td>If set to ‘1’, then this command is part of a sequential read that includes multiple Read commands. If cleared to ‘0’, then no information on sequential access is provided.</td></tr>
<tr><td>05:04</td><td>Access Latency</td><td>
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>None. No latency information provided.</td></tr>
<tr><td>01b</td><td>Idle. Longer latency acceptable.</td></tr>
<tr><td>10b</td><td>Normal. Typical latency.</td></tr>
<tr><td>11b</td><td>Low. Smallest possible latency.</td></tr>
</table>
</td></tr>
<tr><td>03:00</td><td>Access Frequency</td><td>
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>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 50: Read – Command Dword 14

Bits	Description
31:00	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 5.2.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored.

Figure 51: 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 5.2.
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 5.2.

3.2.4.1 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 52.

Figure 52: 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 Field (PRINFO) (refer to Figure 48) settings specified in the command are invalid for the Protection Information with which the namespace was formatted (refer to the PI field in Figure 78 and the DPS field in Figure 97) or the EILBRT field is invalid (refer to section 5.2.2.5).

3.2.5 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 NVMe 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 bit 7 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 bit 7 in the ONCS field 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 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 14, and Command Dword 15 fields.

Figure 53: Verify – Command Dword 2 and Dword 3

Bits	Description
63:48	Reserved

Figure 53: Verify – Command Dword 2 and Dword 3

Bits	Description
47:00	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 5.2.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored.

Figure 54: 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 55: Verify – Command Dword 12

Bits	Description
31	**Limited Retry (LR):** If set to '1', then the controller should apply limited retry efforts. If 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 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 media. There is no implied ordering with other commands. If cleared to '0', then this bit has no effect.
29:26	**Protection Information Field (PRINFO):** Specifies the protection information action and check field, as defined in Figure 9. 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 10.
23:16	Reserved
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.

Figure 56: Verify – Command Dword 14

Bits	Description
31:00	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 5.2.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored.

Figure 57: 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 5.2.

Figure 57: Verify – Command Dword 15

Bits	Description
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 5.2.

3.2.5.1 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). Refer to the Status Code – Status Code Type Values Figure and to Figure 55.

Verify command specific status values are defined in Figure 58.

Figure 58: Verify – Command Specific Status Values

Value	Description
81h	**Invalid Protection Information:** The Protection Information Field (PRINFO) (refer to Figure 55) settings specified in the command are invalid for the Protection Information with which the namespace was formatted (refer to the PI field in Figure 78 and the DPS field in Figure 97) or the EILBRT field is invalid (refer to section 5.2.2.5).

3.2.6 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 59: 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 NVMe Base Specification for the definition of this field.

Figure 60: 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 NVMe Base Specification for the definition of this field.

Figure 61: Write – Command Dword 2 and Dword 3

Bits	Description
63:48	Reserved

Figure 61: Write – Command Dword 2 and Dword 3

Bits	Description
47:00	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 5.2.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored.

Figure 62: 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 63: Write – Command Dword 12

Bits	Description
31	**Limited Retry (LR):** If set to '1', the controller should apply limited retry efforts. If cleared to '0', the controller should apply all available error recovery means to write the data to the NVM.
30	**Force Unit Access (FUA):** If 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 media before indicating command completion. There is no implied ordering with other commands. If cleared to '0', then this bit has no effect.
29:26	**Protection Information Field (PRINFO):** Specifies the protection information action and check field, as defined in Figure 9.
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 10.
23:20	**Directive Type (DTYPE)**: Specifies the Directive Type associated with the Directive Specific field (refer to the Directives section in the NVMe Base Specification).
19:16	Reserved
15:00	**Number of Logical Blocks (NLB):** This field indicates the number of logical blocks to be written. This is a 0's based value.

Figure 64: Write – 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 NVMe Base Specification).
15:08	Reserved

Figure 64: Write – Command Dword 13

<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>Attribute</th><th>Definition</th></tr>
<tr><td>07</td><td>Incompressible</td><td>If set to ‘1’, then data is not compressible for the logical blocks indicated. If cleared to ‘0’, then no information on compression is provided.</td></tr>
<tr><td>06</td><td>Sequential Request</td><td>If set to ‘1’, then this command is part of a sequential write that includes multiple Write commands. If cleared to ‘0’, then no information on sequential access is provided.</td></tr>
<tr><td>05:04</td><td>Access Latency</td><td>
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>None. No latency information provided.</td></tr>
<tr><td>01b</td><td>Idle. Longer latency acceptable.</td></tr>
<tr><td>10b</td><td>Normal. Typical latency.</td></tr>
<tr><td>11b</td><td>Low. Smallest possible latency.</td></tr>
</table>
</td></tr>
<tr><td>03:00</td><td>Access Frequency</td><td>
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>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 65: Write – Command Dword 14

Bits	Description
31:00	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 5.2.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored.

Figure 66: 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 5.2.
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 5.2.

3.2.6.1 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 are defined in Figure 67.

Figure 67: Write – Command Specific Status Values

Value	Description
20h	**Namespace is Write Protected:** The command is prohibited while the namespace is write protected (refer to the Namespace Write Protection section of the NVMe Base Specification).
80h	**Conflicting Attributes:** The attributes specified in the command are conflicting.
81h	**Invalid Protection Information:** The Protection Information Field (PRINFO) (refer to Figure 63) settings specified in the command are invalid for the Protection Information with which the namespace was formatted (refer to the PI field in Figure 78 and the DPS field in Figure 97) or the ILBRT field is invalid (refer to section 5.2.2.5).
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 NVMe Base Specification).

3.2.7 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 bit 1 in the Optional NVM Command Support field in the Identify Controller data structure is set to '1', 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 bit 1 in the ONCS field 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 68: 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 69: Write Uncorrectable – Command Dword 12

Bits	Description
31:16	Reserved
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.

3.2.7.1 Command Completion

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

Figure 70: Write Uncorrectable – Command Specific Status Values

Bit	Description
20h	**Namespace is Write Protected:** The command is prohibited while the namespace is write protected (refer to the Namespace Write Protection section of the NVMe Base Specification).
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 NVMe Base Specification).

3.2.8 Write Zeroes command

The Write Zeroes command is used to set a range of logical blocks 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 bit (PRACT) is cleared to '0', then the protection information for this command shall be all zeroes. If the Protection Information Action bit (PRACT) 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 97), CDW15.LBATM, CDW15.LBAT, as well as CDW2/3 and CDW14 content as described in section 5.2.1.4.1.

After successful completion of this command, the value returned by subsequent reads of logical blocks in this range shall be all bytes cleared to 0h until a write occurs to this LBA range.

If the Deallocate bit (CDW12.DEAC) is set to '1' in a Write Zeroes command, and the namespace supports clearing all bytes to 0h in the values read (e.g., bits 2:0 in the DLFEAT field are set to 001b) from a deallocated logical block and its metadata (excluding protection information), then for each specified logical block, the controller:

- should deallocate that logical block;
- shall return all bytes cleared to 0h in the values read from:
    - that logical block; and
    - that logical blocks metadata (excluding protection information);

and

- shall return the protection information in that logical block as specified in section 3.2.3.2.1.

If the Deallocate bit is cleared to '0' in a Write Zeroes command, and the namespace supports clearing all bytes to 0h in the values read (e.g., bits 2:0 in the DLFEAT field are set to 001b) from a deallocated logical block and its metadata (excluding protection information), then, for each specified logical block, the controller:

- may deallocate that logical block;
- shall return all bytes cleared to 0h in the values read from:
    - that logical block; and
    - that logical blocks metadata (excluding protection information);

and

- shall return the protection information in that logical block based on CDW12.PRINFO in that Write Zeroes command.

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

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

a) if bit 3 in the Optional NVMe Command Support field in the Identify Controller data structure is set to '1', 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 bit 3 in the ONCS field is cleared to '0', then the WZSL field indicates the 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.

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

Figure 71: Write Zeroes – Command Dword 2 and Dword 3

Bits	Description
63:48	Reserved
47:00	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 5.2.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored.

Figure 72: 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.

Figure 73: Write Zeroes – Command Dword 12

Bits	Description
31	**Limited Retry (LR):** If set to '1', the controller should apply limited retry efforts. If cleared to '0', the controller should apply all available error recovery means to write the data to the NVM.
30	**Force Unit Access (FUA):** If set to '1', then the controller shall write the data, and metadata, if any, to non-volatile media before indicating command completion. There is no implied ordering with other commands. If cleared to '0', then this bit has no effect.
29:26	**Protection Information Field (PRINFO):** Specifies the protection information action and check field, as defined in Figure 9. The Protection Information Check (PRCHK) field shall be cleared to 000b.
25	**Deallocate (DEAC):** If set to '1', then the host is requesting that the controller deallocate the specified logical blocks. If cleared to '0', then the host is not requesting that the controller deallocate the specified logical blocks.
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 10. This bit shall be cleared to '0'.
23:16	Reserved
15:00	**Number of Logical Blocks (NLB):** This field indicates the number of logical blocks to be written. This is a 0's based value.

Figure 74: Write Zeroes – Command Dword 14

Bits	Description
31:00	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 5.2.1.4.1. If the namespace is not formatted to use end-to-end protection information, then this field is ignored.

Figure 75: 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 5.2.
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 5.2.

3.2.8.1 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.

Write Zeroes command specific status values are defined in Figure 76.

Figure 76: Write Zeroes – Command Specific Status Values

Value	Description
20h	**Namespace is Write Protected:** The command is prohibited while the namespace is write protected (refer to the Namespace Write Protection section of the NVMe Base Specification).
81h	**Invalid Protection Information:** The Protection Information Field (PRINFO) (refer to Figure 73) settings specified in the command are invalid for the Protection Information with which the namespace was formatted (refer to the PI field in Figure 78 and the DPS field in Figure 97) or the ILBRT field is invalid (refer to section 5.2.2.5).
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 NVMe 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 NVMe 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 NVMe Base Specification. In addition to the Asynchronous Events defined in the NVMe Base Specification, the NVM Command Set defines the Asynchronous Events defined in this section.

Figure 77: Asynchronous Event Information – Notice

Value	Description
00h	**Namespace Attribute Changed:** The Namespace Attribute Changed event operates as defined in the NVMe Base Specification with the following modifications. A controller shall not send this event if: a) Namespace Utilization (refer to Figure 97) has changed, as this is a frequent event that does not require action by the host; b) capacity information (i.e., the NUSE field and the NVMCAP field) returned in the Identify Namespace data structure (refer to Figure 97) 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 5.8.1). 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.

4.1.2 Format NVM command

The Format NVM command operates as defined in the NVMe Base Specification. The Format Index indicates a valid User Data Format from the LBA Format field in the Identify Namespace data structure. Other NVM Command Set specific fields are defined in Figure 78.

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 97) and the namespace capacity (i.e., NCAP) (refer to Figure 97) 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 NVMe Base Specification), then the controller aborts a Format NVM command with a status code of Invalid Namespace or Format that specifies a format (refer to section 5.2.1) of:

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.

Figure 78: 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 set to ‘1’ and protection information is enabled (refer to section 5.2), then protection information is transferred as the first bytes of metadata. If 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 field 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>Protection information is enabled, Type 1</td></tr>
<tr><td>010b</td><td>Protection information is enabled, Type 2</td></tr>
<tr><td>011b</td><td>Protection information is enabled, Type 3</td></tr>
<tr><td>100b 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, Write Zeroes 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 79 defines the Features support requirements for I/O Controllers supporting the NVM Command Set.

Figure 79: Feature Identifiers – NVM Command Set

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

NOTES:
1. This column is only valid if the feature is not saveable (refer to the NVME 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 80 defines the Set Features command specific status values that are specific to the NVM Command Set specific Feature Identifiers used during Command Completion.

Figure 80: Set Features – Command Specific Status Values

Value	Description
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), (Optional)

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 83. 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 82 are returned in Dword 0 of the completion queue entry and the LBA Range Type data structure specified in Figure 83 is returned in the data buffer for that command.

Figure 81: 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 for the Get Features command for this Feature.

Figure 82: 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 83). This is a 0's based value.

Each entry in the LBA Range Type data structure is defined in Figure 83. 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 should return an error 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 should 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 83: 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>Description</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 / 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: Specifies attributes of the LBA range. Each bit defines an attribute.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>0</td><td>If set to ‘1’, the LBA range may be overwritten. If cleared to ‘0’, the area should not be overwritten.</td></tr>
<tr><td>1</td><td>If set to ‘1’, the LBA range should be hidden from the OS / EFI / BIOS. If cleared to ‘0’, the area should be visible to the OS / EFI / BIOS.</td></tr>
<tr><td>2 to 7</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>15:02</td><td>Reserved</td></tr>
<tr><td>23:16</td><td>Starting LBA (SLBA): This field specifies the 64-bit logical block address of the first logical block that is part of this LBA range.</td></tr>
<tr><td>31:24</td><td>Number of Logical Blocks (NLB): This field specifies the number of logical blocks that are part of this LBA range. This is a 0’s based value (e.g., the value 0h specifies one block).</td></tr>
<tr><td>47:32</td><td>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.</td></tr>
<tr><td>63:48</td><td>Reserved</td></tr>
</table>

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

Figure 84: Error Recovery – Command Dword 11

Bits	Description
31:17	Reserved

Figure 84: Error Recovery – Command Dword 11

Bits	Description
16	**Deallocated or Unwritten Logical Block Error Enable (DULBE):** If set to '1', then the Deallocated or Unwritten Logical Block error is enabled for the namespace specified in the NSID field. If cleared to '0', then the Deallocated or Unwritten Logical Block error is disabled for the namespace specified in the NSID field. 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 3.2.3.2.1.
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 LBA Status Information Attributes (Feature Identifier 15h)

The LBA Status Information Poll Interval (LSIPI) (refer to Figure 85) 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 85) 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 85).

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:

a) 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

b) 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 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 85: 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.

4.1.3.4 Host Behavior Support (Feature Identifier 16h)

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

Figure 86: 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 EBLAS field in the Identify Controller data structure in the NVMe Base Specification). If this field is set to 1h and the ELBAS field is set to '1', then the controller: 1) shall report a maximum number that is less than or equal to 64 for: a. the number of LBA formats (refer to the NLBAF field in the Identify Namespace data structure in Figure 97); and b. the number of namespace granularity descriptors (refer to Figure 104); and 2) is enabled to create, format, and perform I/O commands on namespaces formatted with (refer to section 5.2.1): 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, the extended LBA formats (refer to Figure 101) define the actual protection information formats supported. If this field is cleared to 0h, then the controller: 1) shall report a maximum that is less than or equal to 16 for: a. the number of LBA formats; and b. the number of namespace granularity descriptors; 2) shall not create, format, and perform I/O commands on namespaces formatted with (refer to section 5.2.1): 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, and commands requesting these restrictions shall be aborted with a status code of Invalid Namespace or Format. All values other than 0h and 1h are reserved.

4.1.3.5 Write Atomicity Normal (Feature Identifier 0Ah)

This Feature controls the operation of the AWUN and NAWUN parameters (refer to section 2.1.4.1). The attributes are specified in Command Dword 11.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 87 are returned in Dword 0 of the completion queue entry for that command.

Figure 87: Write Atomicity Normal – Command Dword 11

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

4.1.3.6 Asynchronous Event Configuration (Feature Identifier 0Bh)

Figure 88**: Asynchronous Event Configuration – NVM Command Set specific Bit Definitions**

Bits	Description
13	**LBA Status Information Notices:** This bit determines whether an asynchronous event notification is sent to the host for an LBA Status Information Alert event (refer to Figure 77). 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.

4.1.4 Get Log Page command

The Get Log Page command operates as defined in the NVMe Base Specification. In addition to the requirements in the NVMe Base Specification, mandatory, optional, and prohibited Log Identifiers are defined in Figure 89. If a Get Log Page command is processed that specifies a Log Identifier that is not supported, then the controller should abort the command with a status code of Invalid Field in Command.

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

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

Figure 89: Get Log Page – Log Page Identifiers

Log Identifier	Scope	Log Page Name	Reference
01h	Controller	Error Information	4.1.4.1
02h	Controller or NVM Subsystem	SMART / Health Information	4.1.4.2
06h	Controller	Device Self-test	4.1.4.3
0Eh	Controller	LBA Status Information	4.1.4.4

4.1.4.1 Error Information (Log Identifier 01h)

The Error Information Log Page is as defined in the NVMe Base Specification. Figure 90 describes the NVM Command Set specific definition of the User Data field.

Figure 90: Get Log Page – Error Information Log Entry – User Data

Bytes	Description
23:16	**LBA:** This field indicates the first LBA that experienced the error condition, if applicable.

4.1.4.2 SMART / Health Information (02h)

The SMART / Health Information Log Page is as defined in the NVMe Base Specification. For the Data Units Read and Data Units Written fields, when the LBA 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 Identifier 06h)

The Device Self-test Log Page is as defined in the NVMe Base Specification. Figure 91 describes the NVM Command Set specific definition of the Failing User Data Reference field.

Figure 91: Get Log Page – Device Self-test Log

Bytes	Description
23:16	**Failing LBA:** This field indicates the LBA of the logical block that caused the test to fail. If the device encountered more than one failed logical block during the test, then this field only indicates one of those failed logical blocks. The contents of this field are valid only when the FLBA Valid bit is set to '1'.

4.1.4.4 Persistent Event (Log Identifier 0Dh)

The Persistent Event Log Page is as defined in the NVMe Base Specification. Figure 92 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 NVMe Base Specification).

Figure 92: 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 Identify Namespace data structure in the Namespace Management command (refer to Figure 97 and the NVMe Base Specification). For a delete operation that specifies a single namespace this field contains the value from the FLBAS field of the Identify Namespace data (refer to Figure 97) for the namespace being deleted. For a delete operation that specifies all namespaces this field is reserved.
33	**End-to-end Data Protection Type Settings (DPS):** For a create operation, contains the DPS value from the Identify Namespace data structure in the Namespace Management command (refer to Figure 97 and the NVMe Base Specification). For a delete operation that specifies a single namespace this field contains the value from the DPS field of the Identify Namespace data (refer to Figure 97) 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 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. It contains zero or more LBA Status Log Namespace Elements (refer to Figure 94). 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 95). 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 93) 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 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 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 5.8.1.1 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 93: LBA Status Information Log

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 94) contained in this log. 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.
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:** 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 94).

Figure 94: LBA Status Log Namespace Element

Bytes	Description
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 95) returned by the controller in this LBA Status Log Namespace Element. A value of FFFFFFFFh indicates that: a) no LBA Range Descriptors are 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
31:16	**LBA Range Descriptor 0:** This field contains the first LBA Range Descriptor in this LBA Status Log Namespace Element, if present.

Figure 94: LBA Status Log Namespace Element

Bytes	Description
47:32	**LBA Range Descriptor 1:** This field contains the second LBA Range Descriptor in this LBA Status Log Namespace Element, if present.
…	…
(N*16+31): (N*16+16)	**LBA Range Descriptor N:** This field contains the N+1 LBA Range Descriptor in this LBA Status Log Namespace Element, if present.

Figure 95: 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 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.

4.1.5 Identify Command

This specification implements the Identify Command and associated Identify data structures defined in the NVMe 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 NVMe Base Specification. The NVM Command Set is assigned a CSI value of 00h.

Figure 96: CNS Values

CNS Value	O/M [1]	Definition	NSID [2]	CNTID [3]	CSI [4]	Reference Section
Active Namespace Management						
00h	M	Identify I/O Command Set Specific Namespace data structure for the controller processing the command. [6]	Y	N	Y	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
06h	M	Identify I/O Command Set Specific Controller data structure for the controller processing the command. [6]	Y	N	Y	4.1.5.4
11h	O	Identify Namespace data structure for the specified allocated NSID.	Y	N	N	4.1.5.5
16h	O	A Namespace Granularity List (refer to Figure X) is returned to the host for up to sixteen Namespace Granularity Entries.	N	N	N	4.1.5.6

CNS Value	O/M [1]	Definition	NSID [2]	CNTID [3]	CSI [4]	Reference Section

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 NVMe Base Specification).
6. Selection of a UUID may be supported. Refer to the Universally Unique Identifiers (UUIDs) for Vendor Specific Information section in the NVMe 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 97) is returned to the host for that specified namespace. If that specified namespace is an inactive NSID, then the controller returns a zero filled data structure. If the specified namespace is not associated with an I/O Command Set that supports this data structure, then the controller shall abort the command with the status code of Invalid I/O Command Set.

If the controller supports the Namespace Management capability (refer to the Namespace Management section in the NVMe Base Specification) and the NSID field is set to FFFFFFFFh, then the controller returns an Identify Namespace data structure that specifies NVM Command Set capabilities that are common across namespaces for the controller. If the controller does not support the Namespace Management capability and the NSID field is set to FFFFFFFFh, then the controller shall abort the command with a status code of Invalid Namespace or Format.

Figure 97: Identify – Identify Namespace Data Structure, NVM Command Set

Bytes	O/M [1]	Description
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 (*n* - 1). The number of logical blocks is based on the formatted LBA size.
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 LBA 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.
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 Namespace Capacity. The number of logical blocks is based on the formatted LBA size. Refer to section 2.1.1 for details on the usage of this field.

Figure 97: Identify – Identify Namespace Data Structure, NVM Command Set

Bytes	O/M [1]	Description
24	M	**Namespace Features (NSFEAT):** This field defines features of the namespace. Bits 7:5 are reserved. Bit 4 (**OPTPERF**) if set to '1' indicates that the fields NPWG, NPWA, NPDG, NPDA, and NOWS are defined for this namespace and should be used by the host for I/O optimization (refer to section 5.8.2). If cleared to '0', then the controller does not support the fields NPWG, NPWA, NPDG, NPDA, and NOWS for this namespace. Bit 3 (**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 NVMe Base Specification). Bit 2 (**DAE**) if set to '1' indicates that the controller supports the Deallocated or Unwritten Logical Block error for this namespace. If cleared to '0', then the controller does not support the Deallocated or Unwritten Logical Block error for this namespace. Refer to section 3.2.3.2.1. Bit 1 (**NSABP**) if set to '1' indicates that the fields NAWUN, NAWUPF, and NACWU are defined for this namespace and should be used by the host for this namespace instead of the AWUN, AWUPF, and ACWU fields in the Identify Controller data structure. If cleared to '0', then the controller does not support the fields NAWUN, NAWUPF, and NACWU for this namespace. In this case, the host should use the AWUN, AWUPF, and ACWU fields defined in the Identify Controller data structure in the NVMe Base Specification. Refer to section 2.1.4. Bit 0 (**THINP**) if set to '1' indicates that the namespace supports thin provisioning. If cleared to '0' indicates that thin provisioning is not supported Refer to section 2.1.1 for details on the usage of this bit.
25	M	**Number of LBA Formats (NLBAF):** This field defines the number of supported LBA data size and metadata size combinations supported by the namespace. LBA formats shall be allocated in order (starting with 0) and packed sequentially. This is a 0's based value. The maximum number of LBA formats that may be indicated as supported is: a) 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 NVMe Base Specification); or b) 64 if the LBAFEE field is set to 1h in the Host Behavior Support feature (refer to the Host Behavior Support section in the NVMe Base Specification). The supported LBA formats are indicated in bytes 128 to 383 in this data structure. The LBA Format fields with an index beyond the value set in this field are invalid and not supported. LBA Formats that are valid, but not currently available may be indicated by setting the LBA Data Size for that LBA Format to 0h. The metadata may be either transferred as part of the LBA (creating an extended LBA which is a larger LBA size that is exposed to the application) or may be transferred as a separate contiguous buffer of data. The metadata shall not be split between the LBA and a separate metadata buffer. It is recommended that software and controllers transition to an LBA size that is 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 5.8.3.

Figure 97: Identify – Identify Namespace Data Structure, NVM Command Set

<table>
<tr><th>Bytes</th><th>O/M [1]</th><th>Description</th></tr>
<tr><td>26</td><td>M</td><td>Formatted LBA Size (FLBAS): This field indicates the LBA data size & metadata size combination that the namespace has been formatted with (refer to section 4.1.2).

Bits 7 is reserved.

Bits 6:5 indicate the most significant 2 bits of the Format Index of the supported LBA Format indicated in this data structure that was used to format the namespace. If the NLBAF field is less than or equal to 16, then the host should ignore these bits.

Bit 4 if set to ‘1’ indicates that the metadata is transferred at the end of the data LBA, creating an extended data LBA. Bit 4 if cleared to ‘0’ indicates that all of the metadata for a command is transferred as a separate contiguous buffer of data. Bit 4 is not applicable when there is no metadata.

Bits 3:0 indicate the least significant 4 bits of the Format Index of the supported LBA Format indicated in this data structure that was used to format the namespace.</td></tr>
<tr><td>27</td><td>M</td><td>Metadata Capabilities (MC): This field indicates the capabilities for metadata.

Bits 7:2 are reserved.

Bit 1 if set to ‘1’ indicates the namespace supports the metadata being transferred as part of a separate buffer that is specified in the Metadata Pointer. Bit 1 if cleared to ‘0’ indicates that the namespace does not support the metadata being transferred as part of a separate buffer.

Bit 0 if set to ‘1’ indicates that the namespace supports the metadata being transferred as part of an extended data LBA. Bit 0 if cleared to ‘0’ indicates that the namespace does not support the metadata being transferred as part of an extended data LBA.</td></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 5.2.

<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 set to ‘1’ indicates that the namespace supports protection information transferred as the last bytes of metadata. If cleared to ‘0’ indicates that 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 set to ‘1’ indicates that the namespace supports protection information transferred as the first bytes of metadata. If cleared to ‘0’ indicates that the namespace does not support protection information transferred as the first bytes of metadata. For implementations compliant to 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 set to ‘1’ indicates that the namespace supports Protection Information Type 3. If cleared to ‘0’ indicates that the namespace does not support Protection Information Type 3.</td></tr>
<tr><td>1</td><td>Protection Information Type 2 Supported (PIT2S): If set to ‘1’ indicates that 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 set to ‘1’ indicates that the namespace supports Protection Information Type 1. If cleared to ‘0’ indicates that the namespace does not support Protection Information Type 1.</td></tr>
</table>
</td></tr>
</table>

Figure 97: Identify – Identify Namespace Data Structure, NVM Command Set

<table>
<tr><th>Bytes</th><th>O/M [1]</th><th>Description</th></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 5.2.
<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): This bit indicates that the protection information, if enabled, is transferred as the first bytes of metadata. Bit 3 if cleared to ‘0’ indicates that the protection information, if enabled, is transferred as the last bytes of metadata. For implementations compliant to 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>Protection information is enabled, Type 1</td></tr>
<tr><td>010b</td><td>Protection information is enabled, Type 2</td></tr>
<tr><td>011b</td><td>Protection information is enabled, Type 3</td></tr>
<tr><td>100b to 111b</td><td>Reserved</td></tr>
</table></td></tr>
</table></td></tr>
<tr><td>30</td><td>O</td><td>Namespace Multi-path I/O and Namespace Sharing Capabilities (NMIC): This field 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 NVMe Base Specification).</td></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 NVMe Base Specification).</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 NVMe Base Specification).</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.

Bits 7:5 are reserved.

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

Bit 3 if set to ‘1’ indicates that the controller supports the Deallocate bit in the Write Zeroes command for this namespace. If cleared to ‘0’ indicates that 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.

Bits 2:0 indicate 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>

Figure 97: Identify – Identify Namespace Data Structure, NVM Command Set

Bytes	O/M [1]	Description
35:34	O	**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. Refer to section 2.1.4.
37:36	O	**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. Refer to section 2.1.4.
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. Refer to section 2.1.4.
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. Refer to section 2.1.4. Refer to section 5.8.2 for how this field is utilized.
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 5.8.2 for how this field is utilized.
45:44	O	**Namespace Atomic Boundary Size Power Fail (NABSPF):** This field indicates the atomic boundary size for this namespace specific to the Namespace Atomic Write Unit Power Fail value. This field is specified in logical blocks. Writes to this namespace that cross atomic boundaries are not guaranteed to be atomic with respect to other read or write commands and there is no guarantee of data returned on subsequent reads of the associated logical blocks. A value of 0h indicates that there are no atomic boundaries for power fail or error conditions. All other values specify a size in terms of logical blocks using the same encoding as the AWUPF field. Refer to section 2.1.4.
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 5.8.2 for how this field is utilized to improve performance and endurance.

Figure 97: Identify – Identify Namespace Data Structure, NVM Command Set

Bytes	O/M [1]	Description
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 5.3) 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 NVMe Base Specification) or the ANA Persistent Loss state (refer to the ANA Persistent Loss state section in the NVMe Base Specification), then this field shall be cleared to 0h.
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 the OPTPERF bit is cleared to '0', 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 size should be a multiple of Namespace Preferred Write Alignment (NPWA). Refer to section 5.8.2 for how this field is utilized to improve performance and endurance.
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 the OPTPERF bit is cleared to '0', then this field is reserved. The value of this field may change if the namespace is reformatted. Refer to section 5.8.2 for how this field is utilized to improve performance and endurance.
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 the OPTPERF bit is cleared to '0', then this field is reserved. The value of this field may change if the namespace is reformatted. The size should be a multiple of Namespace Preferred Deallocate Alignment (NPDA). Refer to section 5.8.2 for how this field is utilized to improve performance and endurance.
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 the OPTPERF bit is cleared to '0', then this field is reserved. The value of this field may change if the namespace is reformatted. Refer to section 5.8.2 for how this field is utilized to improve performance and endurance.

Figure 97: Identify – Identify Namespace Data Structure, NVM Command Set

Bytes	O/M [1]	Description
73:72	O	**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 the OPTPERF bit is cleared to '0', 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 Write Granularity (NPWG). If the namespace is associated with an NVM set, NOWS defined for this namespace shall be set to the Optimal Write Size field setting defined in NVM Set Attributes Entry (refer to the Namespace Identification Descriptor in the NVMe Base Specification) for the NVM Set with which this namespace is associated. If NOWS is not supported, the Optimal Write Size field in NVM Sets Attributes Entry (refer to the Namespace Identification Descriptor in the NVMe Base Specification) for the NVM Set with which this namespace is associated should be used by the host for I/O optimization. Refer to section 5.8.2 for how this field is utilized to improve performance and endurance.
75:74	O	**Maximum Single Source Range Length (MSSRL):** This field indicates the maximum number of logical blocks that may be specified in the Number of Logical Block field in each valid Source Range Entries Descriptor of a Copy command (refer to section 3.2.2). If the controller supports the Copy command, then this field shall be set to a non-zero value.
79:76	O	**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.
80	O	**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.
91:81		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 NVMe Base Specification).
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 NVMe Base Specification).
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 NVMe Base Specification).
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 NVMe Base Specification).

Figure 97: Identify – Identify Namespace Data Structure, NVM Command Set

Bytes	O/M [1]	Description
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 NVMe 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 NVMe 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 NVMe Base Specification.
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 NVMe 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 NVMe 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 NVMe Base Specification.
LBA Formats		
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 98. Additional information may be provided in the ELBAF0 field (refer to Figure 100).
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 98. Additional information may be provided in the ELBAF1 field (refer to Figure 100).
...		
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 98. Additional information may be provided in the ELBAF63 field (refer to Figure 100).
4095:384	O	Vendor Specific
NOTES: 1. O/M definition: O = Optional, M = Mandatory.		

The LBA format data structure is described in Figure 98.

Figure 98: LBA Format Data Structure, NVM Command Set Specific

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

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 NVMe Base Specification.

Figure 99: 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 with size less than or equal to the AWUN value, the host is guaranteed that the write command is atomic to the NVM with respect to other read or write commands. If a write command is submitted with size greater than the AWUN value, then there is no guarantee of command atomicity. AWUN does not have any applicability to write errors caused by power failure (refer to Atomic Write Unit Power Fail). A value of FFFFh indicates all commands are atomic as this is the largest command size. It is recommended that implementations support a minimum of 128 KiB (appropriately scaled based on LBA size).

Figure 99: Identify – Identify Controller data structure, NVM Command Set Specific Fields

Bytes	O/M [1]	Description
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 with size less than or equal to the AWUPF value, the host is guaranteed that the write is atomic to the NVM with respect to other read or write commands. If a write command is submitted that is greater than this size, there is no guarantee of command atomicity. If the write size is less than or equal to the AWUPF value and the write command fails, then subsequent read commands for the associated logical blocks shall return data from the previous successful write command. If a write command is submitted with size greater than the AWUPF value, then there is no guarantee of data returned on subsequent reads of the associated logical blocks.
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 100 defines the I/O Command Set specific Identify Namespace data structure for the NVM Command Set.

Figure 100: 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></tr>
<tr><td>7:0</td><td>O</td><td>Logical Block Storage Tag Mask (LBSTM): Indicates the mask for the Storage Tag field for the protection information (refer to section 5.2). The size of the mask contained in this field is defined by the STS field (refer to Figure 101). If the size of the mask contained in this field is less than 64 bits, then 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) 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’.</td></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>7:2</td><td>Reserved</td></tr>
<tr><td>1</td><td>16b Guard Protection Information Storage Tag Mask (16BPISTM): If set to ‘1’, then the LBSTM field shall have all bits set to ‘1’ for the 16b Guard Protection Information. If 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.</td></tr>
<tr><td>0</td><td>16b Guard Protection Information Storage Tag Support (16BPISTS): If set to ‘1’, then the end-to-end protection 16b Guard Protection Information format (refer to section 5.2.1.1) supports a non-zero value in the STS field. If 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 97 and Figure 100), then this bit shall be set to ‘1’.</td></tr>
</table></td></tr>
<tr><td>11:9</td><td></td><td>Reserved</td></tr>
<tr><td colspan="3">Extended LBA Format</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 101.</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 101.</td></tr>
<tr><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 101.</td></tr>
<tr><td>4095:268</td><td>O</td><td>Reserved</td></tr>
<tr><td colspan="3">NOTES:
1. O/M definition: O = Optional, M = Mandatory.</td></tr>
</table>

The Extended LBA format data structure is described in Figure 101.

Figure 101: Extended LBA Format Data Structure, NVM Command Set Specific

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:9</td><td>Reserved</td></tr>
<tr><td>8:7</td><td>Protection Information Format (PIF): This field indicates the protection information format (refer to section 5.2.1) when end-to-end protection information is enabled on a namespace formatted with this LBA format.

<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>Reserved</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 5.2.1.4).

This field does limit the minimum and maximum values allowed per protection information formats (refer to section 5.2.1):

<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 102 defines the I/O Command Set Specific Identify Controller data structure for the NVM Command Set.

Figure 102: I/O Command Set Specific Identity Controller Data Structure for the NVM Command Set

Bytes	O/M [1]	Description
00	O	**Verify Size Limit (VSL):** If bit 7 in the Optional NVM Command Support (ONCS) field is set to '1' then: a) a non-zero value in this field indicates the recommended maximum data size for a Verify command (refer to section 3.2.5); and b) a value of 0h in this field indicates that no recommended maximum data size for a Verify command is reported. If bit 7 in the ONCS field 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 3.2.5); 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). This field includes the length of metadata, if metadata is interleaved with the logical block data.
01	O	**Write Zeroes Size Limit (WZSL):** If bit 3 in the Optional NVM Command Support (ONCS) field is set to '1' then: a) a non-zero value in this field indicates the recommended maximum data size for a Write Zeroes command (refer to section 3.2.8); and b) a value of 0h in this field indicates that no recommended maximum data size for a Write Zeroes command is reported. If bit 3 in the ONCS field 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 3.2.8); and b) a value of 0h in this field indicates that the controller does not support the Write Zeroes 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). This field includes the length of metadata, if metadata is interleaved with the logical block data.
02	O	**Write Uncorrectable Size Limit (WUSL):** If bit 1 in the Optional NVM Command Support (ONCS) field is set to '1' then: a) a non-zero value in this field indicates the recommended maximum data size for a Write Uncorrectable command (refer to section 3.2.7); and b) a value of 0h in this field indicates that no recommended maximum data size for a Write Uncorrectable command is reported. If bit 1 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 3.2.7); 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). This field includes the length of metadata, if metadata is interleaved with the logical block data.

03	O	**Dataset Management Ranges Limit (DMRL):** If bit 2 in the Optional NVM Command Support (ONCS) field is set to '1' 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 3.2.3); 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 bit 2 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 3.2.3); 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 bit 2 in the Optional NVM Command Support (ONCS) field is set to '1' 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 3.2.3); 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 bit 2 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 3.2.3); and b) a value of 0h in this field indicates that the controller does not support the Dataset Management command.
15:08	O	**Dataset Management Size Limit (DMSL):** If bit 2 in the Optional NVM Command Support (ONCS) field is set to '1' 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 3.2.3). 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 bit 2 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 3.2.3); and b) a value of 0h in this field indicates that the controller does not support the Dataset Management command.
4095:16	M	Reserved
NOTES: 1. O/M definition: O = Optional, M = Mandatory.		

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

An Identify Namespace data structure (refer to Figure 97) is returned to the host for the namespace specified in the Namespace Identifier (NSID) field if it is an allocated NSID. If the specified namespace is an unallocated NSID, then the controller returns a zero filled data structure.

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

4.1.5.6 Namespace Granularity List (CNS 16h)

If the controller supports reporting of Namespace Granularity (refer to section 5.3), then a Namespace Granularity List (refer to Figure 103) is returned to the host for up to:

a) 16 namespace granularity descriptors (refer to Figure 104) 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 NVMe Base Specification); or
b) 64 namespace granularity descriptors if the LBAFEE field is set to 1h in the Host Behavior Support feature.

Figure 103: Namespace Granularity List

Bytes	Description
03:00	**Namespace Granularity Attributes:** This field indicates attributes of the Namespace Granularity List. Bits 31:1 are reserved. Bit 0 **(Granularity Descriptor Mapping):** If set to '1', then each valid namespace granularity descriptor applies to the LBA format having the same index and the Number of Descriptors field shall be equal to the Number of LBA Formats field in the Identify Namespace data structure (refer to Figure 97). If cleared to '0', then NG Descriptor 0 shall apply to all LBA formats and the Number of Descriptors field shall be cleared to 0h.
04	**Number of Descriptors:** 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.
31:05	Reserved
47:32	**NG Descriptor 0:** 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.
63:48	**NG Descriptor 1:** This field contains the second namespace granularity descriptor in the list. This namespace granularity descriptor applies to LBA Format 1.
…	…
1055:1040	**NG Descriptor 63:** This field contains the sixteenth namespace granularity descriptor in the list. This namespace granularity descriptor applies to LBA Format 63.

The format of the namespace granularity descriptor is defined in Figure 104.

Figure 104: Namespace Granularity Descriptor

Bytes	Description
07:00	**Namespace Size Granularity:** 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:** 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.

4.1.6 Namespace Management command

The Namespace Management command operates as defined in the NVMe 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 105. 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 NVMe 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 5.2.1):

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.

Figure 105: Namespace Management – Host Software Specified Fields

Bytes	Description	Host Specified
Fields that are a subset of the Identify Namespace data structure (refer to Figure 97)		
07:00	Namespace Size (NSZE)	Yes
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]	Yes
103:102	Endurance Group Identifier (ENDGID)	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
511:392	Reserved	

Notes:
1. A value of 0h specifies that the controller determines the value to use (refer to the Namespace Management section of the NVMe Base Specification). If the associated feature is not supported, then this field is ignored by the controller.

4.1.7 Sanitize command – NVM Command Set Specific

The Sanitize command operates as defined in the NVMe Base Specification.

In addition to the requirements in the NVMe Base Specification, the following NVM Command Set admin commands (refer to Figure 106) are allowed if a sanitize operation is in progress:

Figure 106: 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 NVMe Base Specification.

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

4.2 I/O Command Set Specific Admin commands

In addition to the I/O Command Set Specific Admin commands defined in the NVMe 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 Potentially Unrecoverable LBAs (refer to section 5.8.1). If the Get LBA Status command completes successfully, then the LBA Status Descriptor List, defined in Figure 111, 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.

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 discovery of Untracked LBAs, they may or may not be added to the list of Tracked 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 bits 1:0 in the Status field should be set to 11b in that entry.

Figure 107: 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 NVMe Base Specification for the definition of this field.

Figure 108: 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 109: 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 110: Get LBA Status – Command Dword 13

Bits	Description
31:24	**Action Type (ATYPE):** This field specifies the mechanism the controller uses in determining the LBA Status Descriptors to return as defined in Figure 112. **Value** / **Description** 10h: Perform a scan and return Untracked LBAs and Tracked LBAs in the specified range 11h: Return Tracked LBAs in the specified range All others: Reserved
23:16	Reserved
15:00	**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 97).

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 namespace specified in the Namespace Identifier (CDW1.NSID);
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 namespace specified in the Namespace Identifier (CDW1.NSID) field;
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.

Figure 111: LBA Status Descriptor List

Bytes	Description
03:00	**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 Entries are returned.

Figure 111: LBA Status Descriptor List

<table>
<tr><th>Bytes</th><th>Description</th></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>Code</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 ATYPE set to 10h or ATYPE set to 11h, 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><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>(N*16+23): (N*16+8)</td><td>LBA Status Descriptor Entry N: This field contains the N+1 LBA Status Descriptor Entry in the list, if present.</td></tr>
</table>

Figure 112: 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 112: LBA Status Descriptor Entry

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>13</td><td>Status: This field contains information about this LBA range.
<table>
<tr><th>Bits</th><th>Definition</th></tr>
<tr><td>7:2</td><td>Reserved</td></tr>
<tr><td>1:0</td><td>These bits indicate information about the logical blocks indicated in this LBA Status Descriptor Entry.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</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></tr>
<tr><td>01b</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></tr>
<tr><td>10b</td><td>Reserved</td></tr>
<tr><td>11b</td><td>Each logical block is a:
logical blosck for which the most recent write to the logical block was a Write Uncorrectable command.</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 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.

4.2.1.1 Command Completion

When the command is completed, the controller posts a completion queue entry to the Admin Completion Queue indicating the status for the command.

5 Extended Capabilities

5.1 Asymmetric Namespace Access Reporting

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

Figure 113 describes Asymmetric Namespace Access effects on command processing that are specific to the NVM Command Set and its associated Feature Identifiers.

Figure 113: 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 97) 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 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 host software 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. If the Metadata Size (refer to Figure 101) 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 5.8.3, 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 148), while DIX functionality is achieved by configuring the metadata and data to be in separate buffers (as shown in Figure 149).

The NVM Express interface supports the same end-to-end protection types defined in the SCSI Protection information model specified in SBC-3. 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 97).

5.2.1 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.

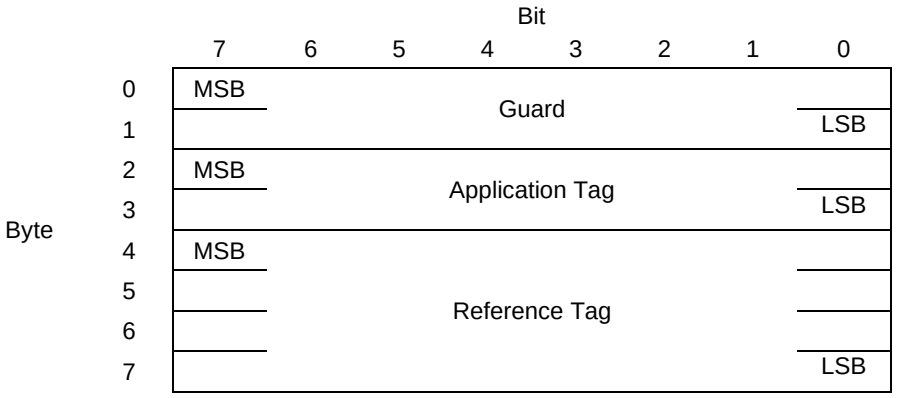
The following protection information formats are defined as extended protection information formats and are only supported when the Controller Attributes (CTRATT) field has the Extended LBA Formats Supported (ELBAS) bit set to ‘1’ (refer to the Identify Controller data structure in the NVMe Base Specification):

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.

5.2.1.1 16b Guard Protection Information

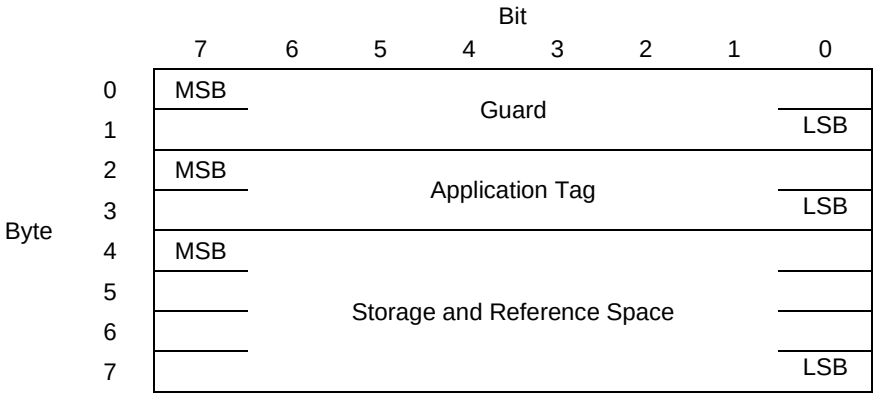
If the Storage Tag Size (STS) field for the LBA Format is cleared to 0h, then the 16b Guard Protection Information is shown in Figure 114 and is contained in the metadata associated with each logical block. The Guard field contains a CRC-16 computed over the logical block data. The formula used to calculate the CRC-16 is defined in SBC-3. 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 114: 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 115. The Storage and Reference Space field is separated into a Storage Tag field and a Logical Block Reference Tag field as defined in section 5.2.1.4. 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 115: 16b Guard Protection Information Format when STS field is non-zero

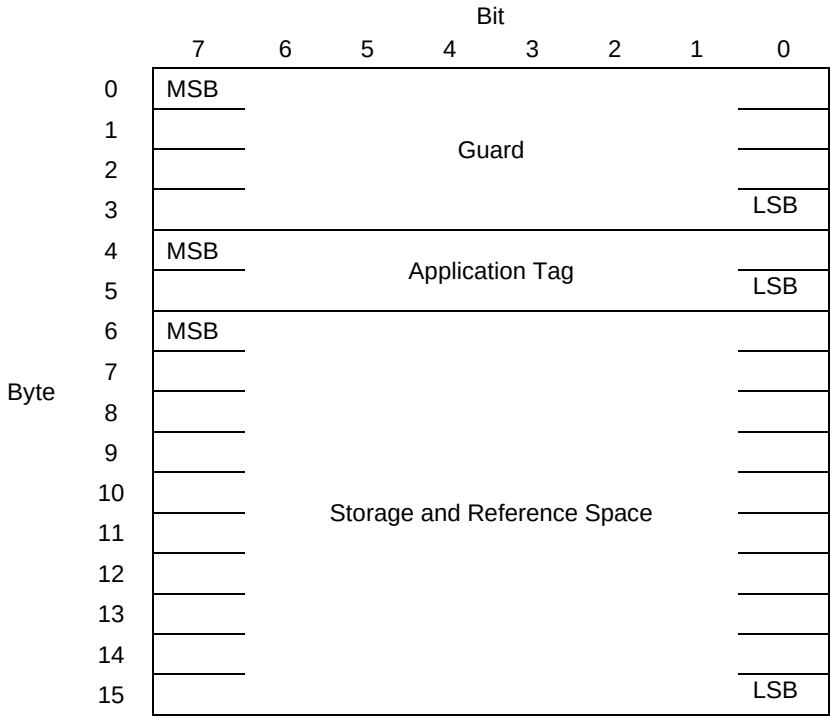


5.2.1.2 32b Guard Protection Information

The 32b Guard Protection Information is shown in Figure 116 and is contained in the metadata associated with each logical block. The 32b Guard Protection Information shall only be available to namespaces that have an LBA size (refer to the LBADS field in Figure 98) 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 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 5.2.1.1).

Figure 116: 32b Guard Protection Information Format



5.2.1.2.1 32b CRC Test Cases

Several 32b CRC test cases are shown in Figure 117.

Figure 117: 32b CRC Test Cases for 4 KiB Logical Block with no Metadata

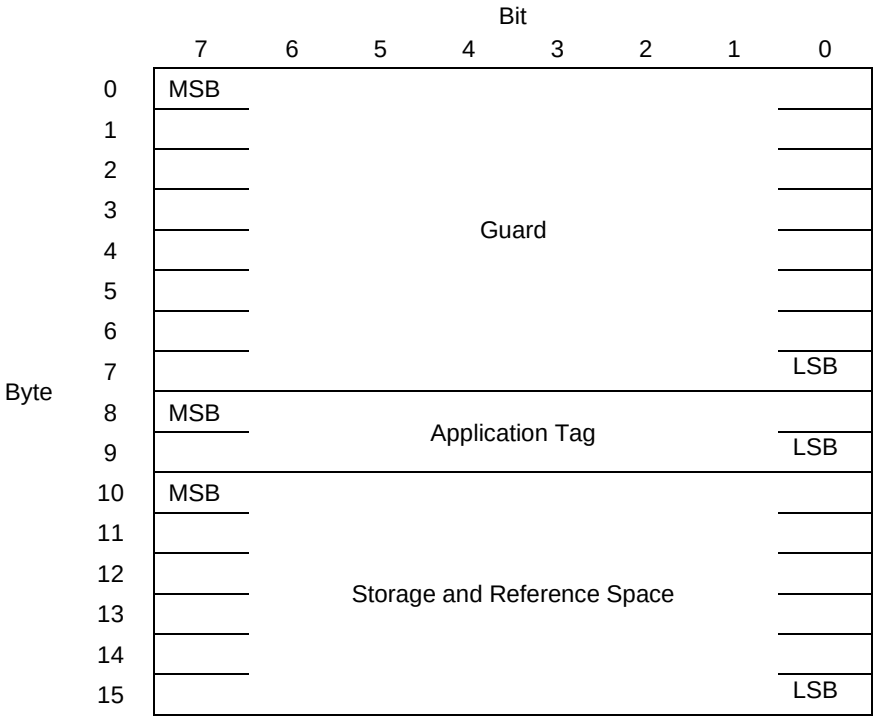
Logical Block Contents	32b Guard Field Value
4 KiB bytes each byte cleared to 00h	98F94189h
4 KiB bytes each byte set to FFh	25C1FE13h
4 KiB bytes of an incrementing byte pattern from 00h to FFh, repeating (e.g. byte 0 is set to 00h, byte 1 is set to 01h, … , byte 254 is set to FEh, byte 255 is set to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

5.2.1.3 64b Guard Protection Information

The 64b Guard Protection Information is shown in Figure 118 and is contained in the metadata associated with each logical block. 64b Guard Protection Information shall only be available to namespaces that have an LBA size (refer to the LBADS field in Figure 98) 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 Figure 118. The Application Tag and Storage and Reference Space have the same definition as defined by 16b Guard Protection Information (refer to section 5.2.1.1).

Figure 118: 64b Guard Protection Information Format



5.2.1.3.1 64b CRC Definition

Figure 119 defines the 64b CRC polynomial used to generate the Guard field for the 64b Guard Protection Information.

Figure 119: 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.

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.

5.2.1.3.2 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)):

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

5.2.1.3.3 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.

5.2.1.3.4 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 following Rocksoft™ Model CRC Algorithm parameters:

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 118 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 120: 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.

5.2.1.3.5 64b CRC Test Cases

Several 64b CRC test cases are shown in Figure 121.

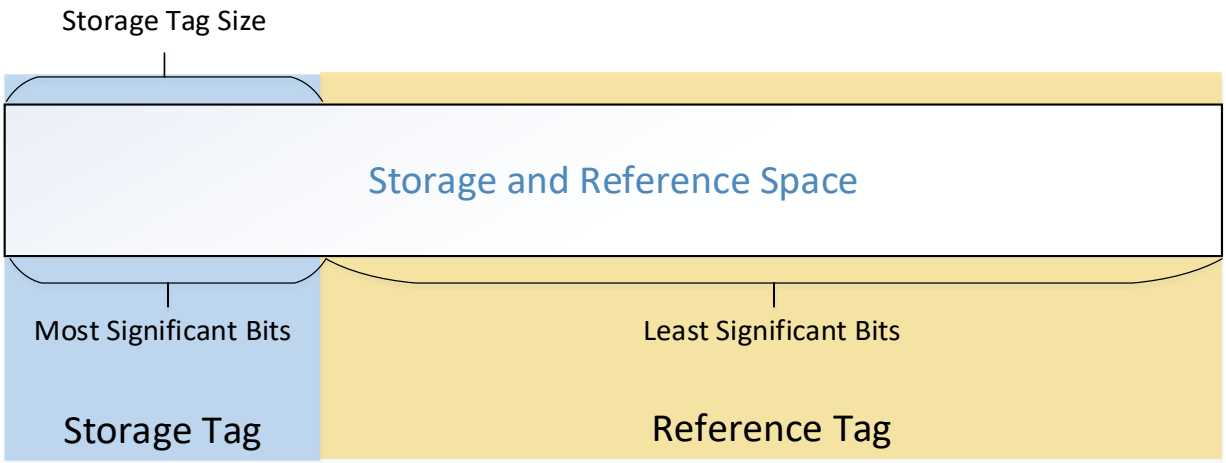
Figure 121: 64b CRC Test Cases for 4 KiB Logical Block with no Metadata

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

5.2.1.4 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 122. 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 122: Separation of Storage and Reference Space into Storage Tag and Logical Block Reference Tag



5.2.1.4.1 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 123 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 101) for each protection information format (refer to section 5.2.1).

Figure 123: 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 124 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 (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 124: LBST, ELBST, ILBRT, and EILBRT fields Format in Command Dwords

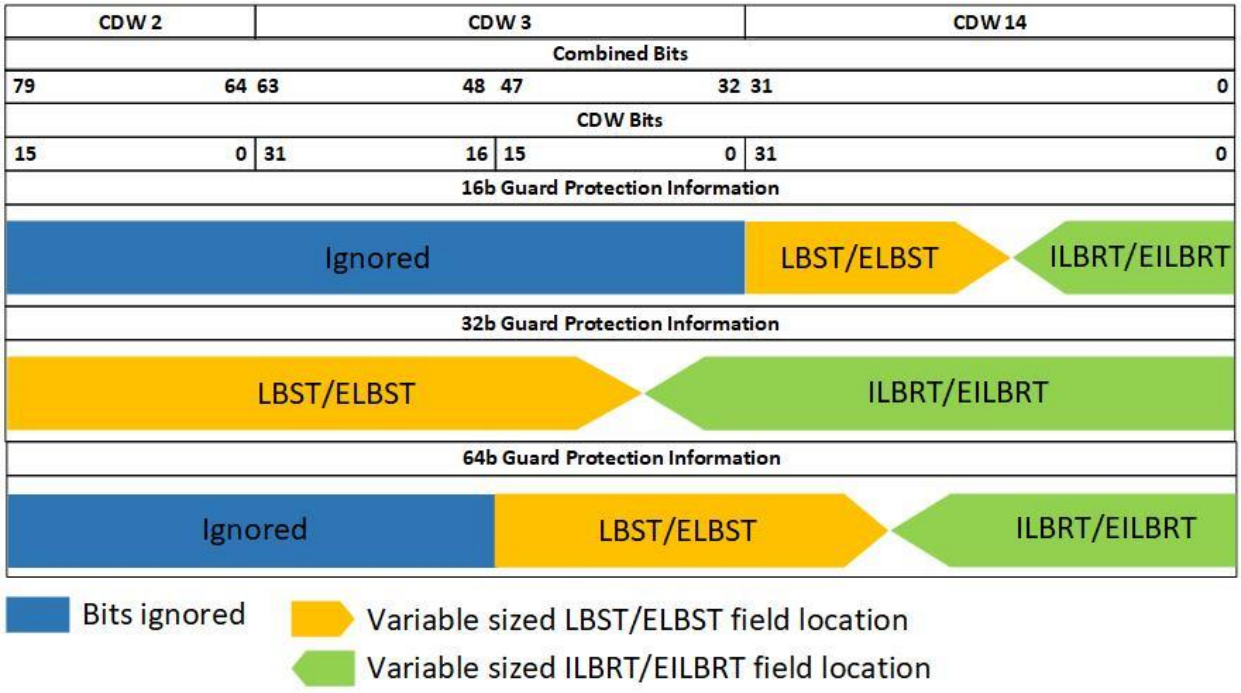


Figure 125 shows the layout of the LBST, ELBST, ILBRT, and EILBRT fields for I/O commands that utilize the fields.

Figure 125: I/O Command LBST, ELBST, ILBRT, and EILBRT fields Format

Bits	Description
16b Guard Protection Information	
Command Dword 2	
15:00	Ignored
Command Dword 3	
31:00	Ignored
Command Dword 14	
31:00	Variable sized LBST, ELBST, ILBRT or EILBRT as defined in Figure 124
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 124
Command Dword 14	
31:00	Variable sized LBST, ELBST, ILBRT or EILBRT as defined in Figure 124
64b Guard Protection Information	
Command Dword 2	
15:00	Ignored
Command Dword 3	
31:16	Ignored
15:00	Variable sized LBST, ELBST, ILBRT or EILBRT as defined in Figure 124
Command Dword 14	
31:00	Variable sized LBST, ELBST, ILBRT or EILBRT as defined in Figure 124

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 101) cleared to 0h specifying a 512B LBA size;
b) Metadata Size field (refer to Figure 101) set to 8h specifying an 8B metadata size;
c) Protection Information Format field (refer to Figure 100) cleared to 00b specifying the 16b Guard Protection Information; and
d) Storage Tag Size (STS) field (refer to Figure 101) 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 126 and for a Read command is shown in Figure 127.

Figure 126: 16b Guard Protection Information Write Command Example

Bits	Description
Command Dword 2	
15:00	Ignored
Command Dword 3	
31:00	Ignored
Command Dword 14	
31:00	ILBRT

Figure 127: 16b Guard Protection Information Read Command Example

Bits	Description
Command Dword 2	
15:00	Ignored
Command Dword 3	
31:00	Ignored
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 101) set to Ch specifying a 4 KiB LBA size;
b) Metadata Size field (refer to Figure 101) set to 10h specifying a 16B metadata size;
c) Protection Information Format field (refer to Figure 100) set to 01b specifying the 32b Guard Protection Information; and
d) Storage Tag Size (STS) field (refer to Figure 101 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 128 and for a Read command is shown in Figure 129.

Figure 128: 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
Command Dword 14	
31:00	Least significant 32 bits of ILBRT

Figure 129: 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 101) set to Ch specifying a 4 KiB LBA size;
b) Metadata Size field (refer to Figure 101) set to 10h specifying a 16B metadata size;
c) Protection Information Format field (refer to Figure 100) set to 10b specifying the 64b Guard Protection Information; and
d) Storage Tag Size (STS) field (refer to Figure 101) 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 130 and for a Read command is shown in Figure 131.

Figure 130: 64b Guard Protection Information Write Command Example

Bits	Description
Command Dword 2	
15:00	Ignored
Command Dword 3	
31:16	Ignored
15:00	Most significant 16 bits of LBST
Command Dword 14	
31:30	Least significant 2 bits of LBST
29:00	ILBRT

Figure 131: 64b Guard Protection Information Read Command Example

Bits	Description
Command Dword 2	
15:00	Ignored
Command Dword 3	
31:16	Ignored
15:00	Most significant 16 bits of the ELBST
Command Dword 14	
31:30	Least significant 2 bits of ELBST
29:00	EILBRT

5.2.2 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.

5.2.2.1 Protection Information and Write Commands

Figure 132 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).

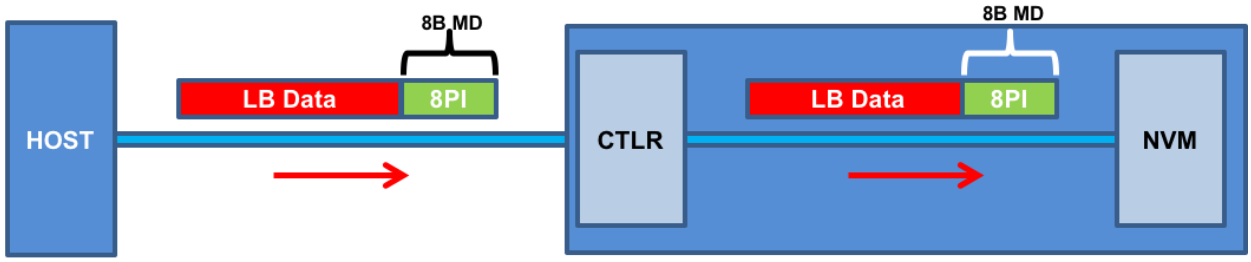
If the namespace is formatted with protection information and the PRACT bit is set to ‘1’, then:

1. If the namespace is formatted with Metadata Size (refer to Figure 98) equal to protection information size (refer to section 5.2.1), 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

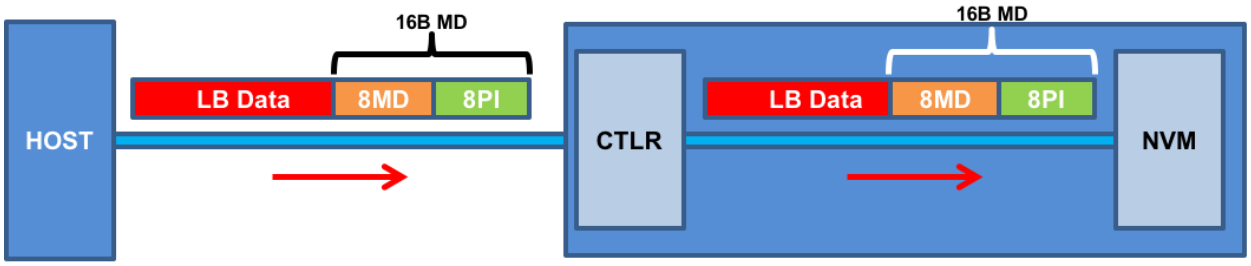
block data and protection information are written to NVM (i.e., the metadata is not resident within the host buffer); and

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. 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 97).

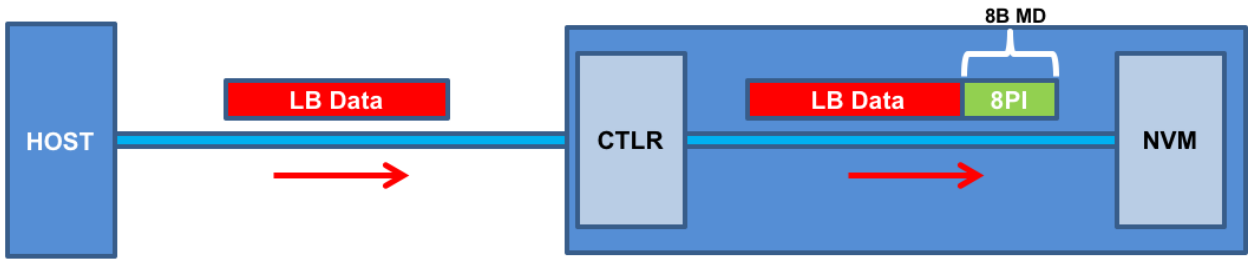
Figure 132: Write Command 16b Guard Protection Information Processing



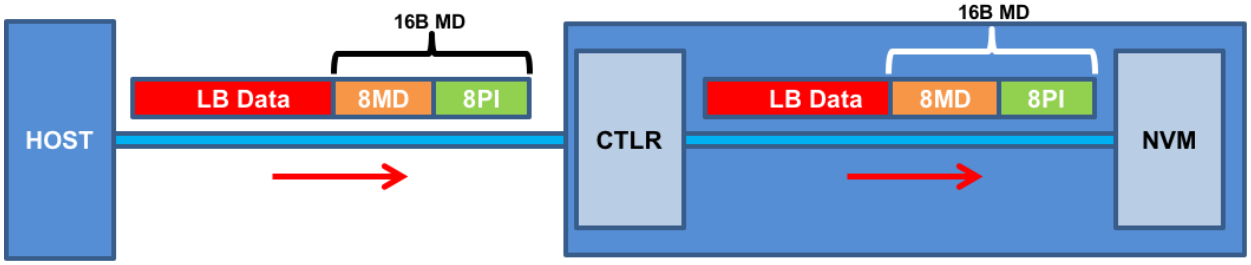
a) MD=8, PI, PRACT=0: Metadata remains same size in NVM and host buffer



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

5.2.2.2 Protection Information and Read Commands

Figure 133 provides some examples of the protection information processing that may occur as a side effect of Read command processing.

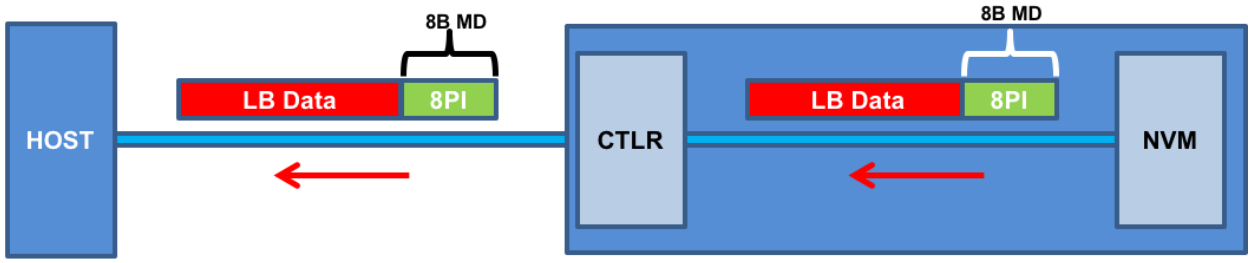
If the namespace is formatted with protection information and the PRACT bit is cleared to ‘0’, then the logical block data and metadata, which in this case contains the protection information and possibly

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).

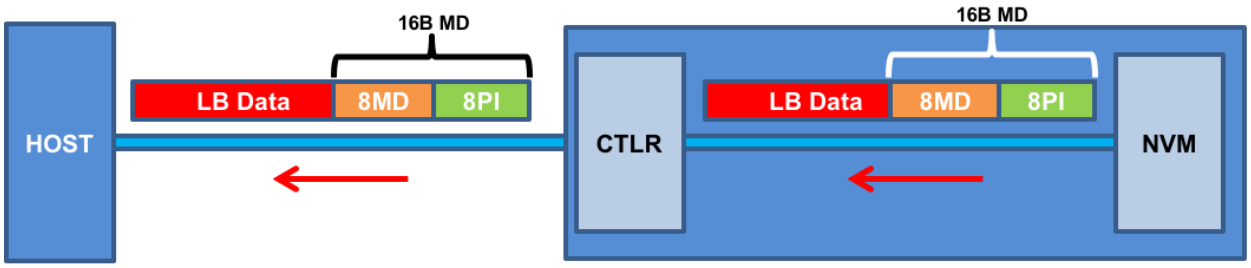
If the namespace is formatted with protection information and the PRACT bit is set to '1', then:

a) If the namespace is formatted with Metadata Size (refer to Figure 98) equal to protection information size (refer to section 5.2.1), 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).

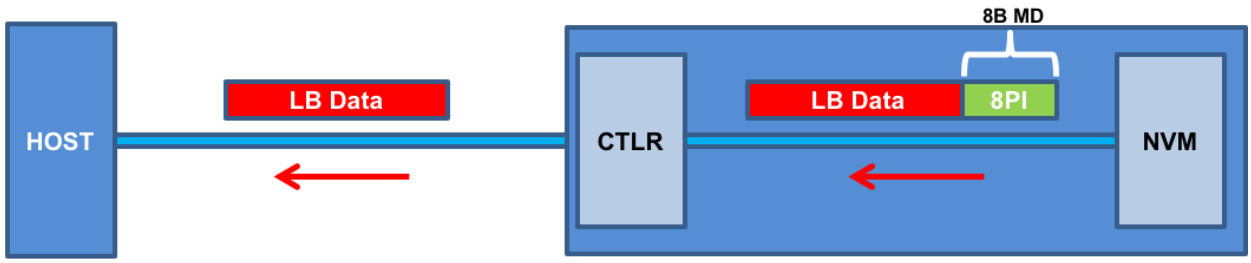
Figure 133: Read 16b Guard Command Protection Information Processing



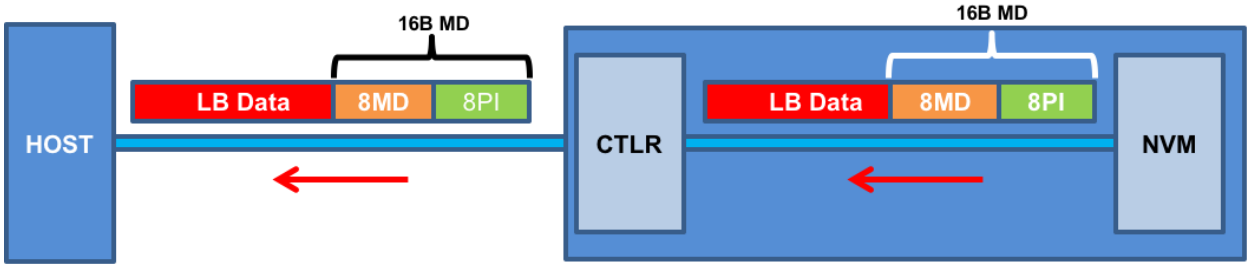
a) MD=8, PI, PRACT=0: Metadata remains same size in NVM and host buffer



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

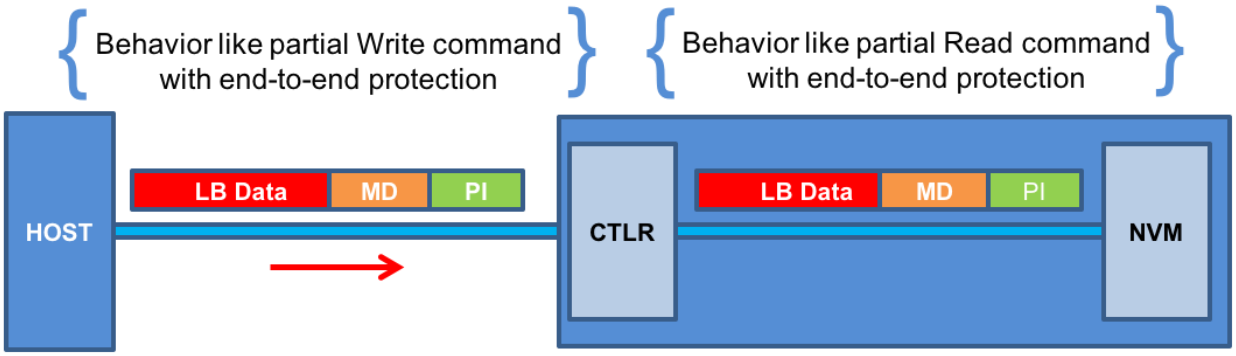
5.2.2.3 Protection Information for Fused Operations

Protection processing for fused operations is the same as those for the individual commands that make up the fused operation.

5.2.2.4 Protection Information and Compare commands

Figure 134 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 5.2.2.1). 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 5.2.2.2).

Figure 134: Protection Information Processing for Compare



Protection Information with PRACT bit set to '0' (i.e., pass)

5.2.2.5 Protection Information and Copy commands

Protection information processing during a Copy command parallels both Write and Read commands. For the portion of the Copy command that transfers data and protection information from the LBAs described by a Source Range Entry (refer to Figure 34), the protection information checks performed by the controller are controlled by the PRINFOR field in Copy command Dword 12 (refer to Figure 30) and parallels the Read command protection information checks (refer to section 5.2.2.2) as follows:

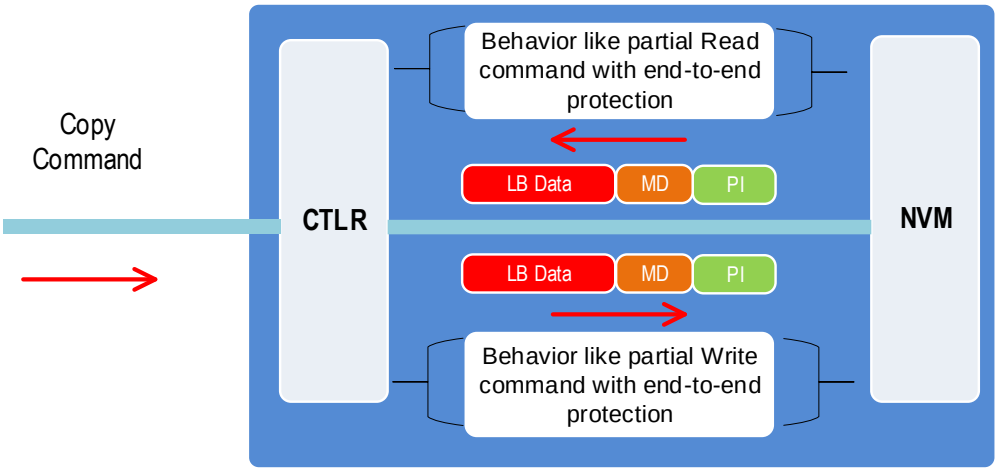
- The logical block data and metadata is transferred from the NVM to the controller.
- As the logical block data and metadata pass through the controller, the protection information within the metadata is checked. If a protection information check error is detected, the command completes with the status 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).

For the portion of the Copy command that transfers data and protection information to the LBAs starting at the SDLBA field (refer to Figure 29), the protection information operations performed by the controller are controlled by the PRINFOW field in Copy command Dword 12 (refer to Figure 30) and parallels the Write command protection information checks (refer to section 5.2.2.1) 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 protection information is handled as described in section 5.2.2.1.

If the PRACT bit is cleared to '0' in the PRINFOR field and the PRACT bit is set to '1' in the PRINFOW field, then the Copy command shall be aborted with a status code of Invalid Field in Command. If the PRACT bit is set to '1' in the PRINFOR field and the PRACT bit is cleared to '0' in the PRINFOW field, then the Copy command shall be aborted with a status code of Invalid Field in Command.

Figure 135: Protection Information Processing for Copy



Protection Information with PRACT bit cleared to '0' (i.e., pass)

While 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, it should be compared to the original guard field (i.e., the guard field associated with the read portion of the copy operation).

5.2.3 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 5.2.1) 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 5.2.1.4) 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 Storage Tag Mask (LBSTM) field in the NVM Command Set Identify Namespace data structure (refer to Figure 100).
- If the Reference Tag is defined (refer to Figure 101) 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 least significant four bytes of the SLBA field.

  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 host software 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 (refer to section 5.2.1.4) 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 should be aborted with a status code of Invalid Protection Information, but may be aborted with a status code of Invalid Field in Command.
- 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 5.2.1) 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.

5.3 Namespace Management

Namespace Management operates as defined in the NVMe 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 content. The Namespace Granularity List defined in Figure 103 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 103) contains one or more Namespace Granularity Descriptors (refer to Figure 104) indicating the size granularity and the capacity granularity at which the controller allocates 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) and Namespace Capacity (NCAP) fields of the data structure used for the create operation of the Namespace Management command (refer to the Namespace Management command section in the NVMe Base Specification).

If a Namespace Management command create operation specifies values such that:

a) the product of NSZE and the Formatted LBA Size value is an integral multiple of the Namespace Size Granularity;
b) the product of NCAP and the Formatted LBA Size value 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).

5.4 NVM Command Set Media and Data Error Handling

Media and Data Error handling operates as described in the NVMe 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.

5.5 Reservations

Reservations operate as defined in the NVMe Base Specification with the additional I/O Command Set specific Command Behavior in the Presence of a Reservation defined in Figure 136.

Figure 136: Command Behavior in the Presence of a Reservation

NVMe Command	Write Exclusive Reservation		Exclusive Access Reservation		Write Exclusive Registrants Only or Write Exclusive All Registrants Reservation		Exclusive Access Registrants Only or Exclusive Access All Registrants Reservation	
	Non-Registrant	Registrant	Non-Registrant	Registrant	Non-Registrant	Registrant	Non-Registrant	Registrant
Copy Command Group								
Copy	C	C	C	C	C	A	C	A
Read Command Group								
Compare Read Verify	A	A	C	C	A	A	C	A
Write Command Group								
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								

5.6 Sanitize Operations

Sanitize operates as defined in the NVMe 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 NVMe Base Specification) of the NVM subsystem are specified in Figure 137. If the controller deallocates user data after successful completion of a sanitize operation, then values read from deallocated logical blocks are described in section 3.2.3.2.1. 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 137: Sanitize Operations – User Data Values

Sanitize Operation	User Data
Block Erase	Vendor specific value
Crypto Erase	Indeterminate
Overwrite	Refer to Sanitize Operations – Overwrite Mechanism in the NVMe Base Specification

5.7 Streams

Streams operate as defined in the NVMe Base Specification. The unit of granularity for the NVM Command Set specific definition of the Stream Write Size (SWS) field is in logical blocks.

5.8 Command Set Specific Capability

5.8.1 Get LBA Status

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 is able to be recovered from another location and re-written.

To support the Get LBA Status capability, the NVM subsystem shall:

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

Prior to using the Get LBA Status capability:

- The host should use the Get Features and Set Features commands with the LBA Status Information Attributes Feature (refer to section 4.1.3.3) 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 88).

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 Identifier 0Eh with the Retain Asynchronous Event bit set to ‘1’ in order to read the entire LBA Status Information log page (refer to section 4.1.4.1).

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 re-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 Identifier 0Eh with the Retain Asynchronous Event bit cleared to '0'. Issuing a Get Log Page command for Log 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 Identifier 0Eh with the Retain Asynchronous Event bit cleared to '0'. Section 5.8.1.1 describes some example host implementations.

The Get LBA Status command may return zero or more LBA Status Descriptors (refer to Figure 111) for each LBA Range Descriptor (refer to Figure 95) returned by the LBA Status Information log page.

Figure 138: Example LBA Range Identifiers returned by LBA Status Information Log Page

Bytes	Description	Value
03:00	Namespace Element Identifier	1
07:04	Number of LBA Range Descriptors	2
15:08	Reserved	
31:16	**LBA Range Descriptor 0:** This field contains the first LBA Range Descriptor in this LBA Status Log Namespace Element.	**Description** / **Value** Range Starting LBA: 10 Range Number of Logical Blocks: 1,000
47:32	**LBA Range Descriptor 1:** This field contains the second LBA Range Descriptor in this LBA Status Log Namespace Element.	**Description** / **Value** Range Starting LBA: 15,000 Range Number of Logical Blocks: 15,010

Figure 139: Example LBA Status Descriptors for Get LBA Status Command issued for LBA Range Descriptor 0 in Figure 138 (Starting LBA set to 10, Range Length set to 1,000)

Bytes	Description	Value
03:00	Number of LBA Status Descriptors	3
04	Completion Condition	2
07:05	Reserved	
23:08	**LBA Range Descriptor 0:** This field contains the first LBA Range Descriptor in this LBA Status Log Namespace Element.	**Description** / **Value** Descriptor Starting LBA: 10 Number of Logical Blocks: 30
39:24	**LBA Range Descriptor 1:** This field contains the second LBA Range Descriptor in this LBA Status Log Namespace Element.	**Description** / **Value** Descriptor Starting LBA: 550 Number of Logical Blocks: 75

Figure 139: Example LBA Status Descriptors for Get LBA Status Command issued for LBA Range Descriptor 0 in Figure 138 (Starting LBA set to 10, Range Length set to 1,000)

Bytes	Description	Value
55:40	**LBA Range Descriptor 2:** This field contains the third LBA Range Descriptor in this LBA Status Log Namespace Element.	**Description** / **Value** Descriptor Starting LBA / 1,000 Number of Logical Blocks / 10

Figure 140: Example LBA Status Descriptors for Get LBA Status Command issued for LBA Range Descriptor 1 in Figure 138 (Starting LBA set to 15,000, Range Length set to 15,010)

Bytes	Description	Value
03:00	**Number of LBA Status Descriptors**	1
04	**Completion Condition**	2
07:05	Reserved	
23:08	**LBA Range Descriptor 0:** This field contains the LBA Range Descriptor in this LBA Status Log Namespace Element.	**Description** / **Value** Descriptor Starting LBA / 15,000 Number of Logical Blocks / 15,010

5.8.1.1 Sample Get LBA Status Host Software Implementations (Informative)

5.8.1.1.1 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’.

5.8.1.1.2 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.

5.8.2 Improving Performance through I/O Size and Alignment Adherence

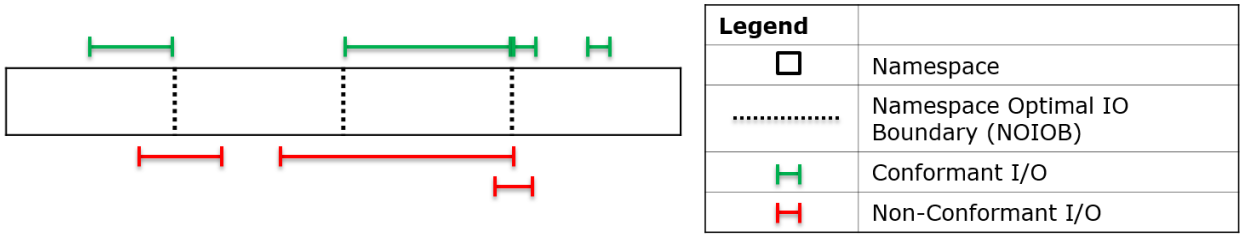
NVMe controllers may require constrained I/O sizes and alignments to achieve the full performance potential. There are a number of optional attributes that the controller uses to indicate these recommendations. If hosts do not follow these constraints, then the controller shall function correctly, but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

Each Copy, Write, Write Uncorrectable, or Write Zeroes commands should address a multiple of Namespace Preferred Write Granularity (NPWG) (refer to Figure 97) and Stream Write Size (SWS) (refer to the Streams Directive – Return Parameters Data Structure figure in the NVMe Base Specification) logical blocks (as expressed in the NLB field), and the SLBA field of the command should be aligned to Namespace Preferred Write Alignment (NPWA) (refer to Figure 97) for best performance. Each range in a Dataset Management command with the Attribute - Deallocate (AD) bit set to '1' should contain a multiple of Namespace Preferred Deallocate Granularity (NPDG) (refer to Figure 97) logical blocks and the start of each range should be aligned to Namespace Preferred Deallocate Alignment (NPDA) (refer to Figure 97) and Stream Granularity Size (SGS) (refer to the Streams Directive – Return Parameters Data Structure figure in the NVMe Base Specification) logical blocks.

5.8.2.1 Improved I/O examples (non-normative)

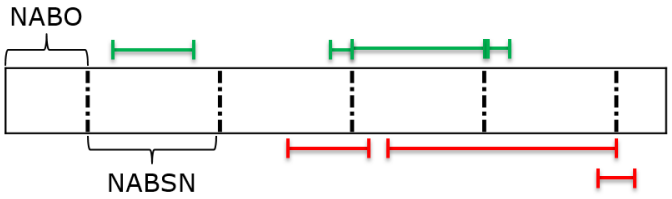
It is recommended that the host utilize the I/O attributes as reported by the controller to receive optimal performance from the NVM subsystem. This section summarizes performance related attributes from namespaces, streams, NVM Sets and the NVM command set. The I/O commands discussed throughout this section include those that interact with non-volatile storage 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 141: An example namespace with four NOIOBs



In Figure 141 an example namespace is diagrammed with three Namespace I/O Boundaries (NOIOB) (refer to Figure 97). The NOIOB attribute should be applied to Read, Compare, Copy, Verify, Write, Write Uncorrectable, and Write Zeroes I/O commands. 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 141 cross an NOIOB. The three I/O commands shown in red on the bottom of Figure 141 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 could be split into two I/O commands that adhere to the recommendations provided by the namespace for NOIOB.

Figure 142: Example namespace illustrating a potential NABO and NABSN

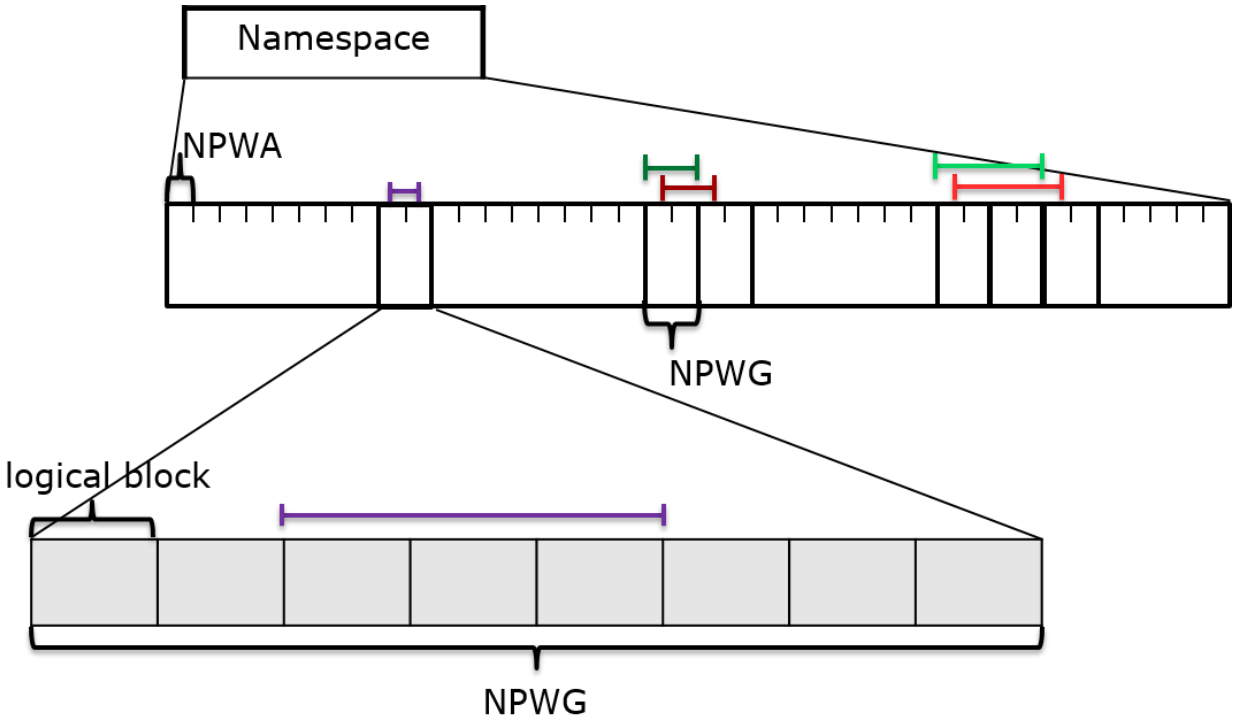


Legend	
	Namespace
	Namespace Atomic Boundary Size Normal (NABSN)
	Conformant I/O
	Non-Conformant I/O
Namespace Atomic Boundary Offset (NABO)	

Continuing with the same namespace example from Figure 141, an illustration of Namespace Atomic Boundary Offset (NABO) (refer to Figure 97) and Namespace Atomic Boundary Size Normal (NABSN) (refer to Figure 97) is shown in Figure 142. 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 142. The I/O commands shown in green on the top of Figure 142 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 141 and Figure 142 both illustrate commands that could 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 141 and Figure 142 could 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.

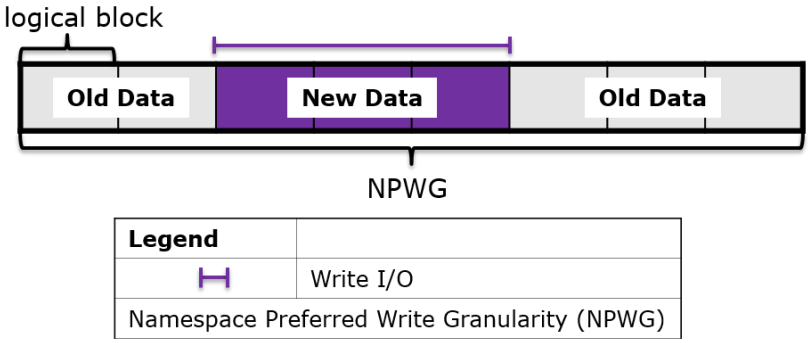
Figure 143: Example namespace broken down to illustrate potential NPWA and NPWG settings



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

NPWG and NPWA are namespace internal constructs, and they are illustrated in Figure 143. The box at the top of Figure 143 is the namespace. The series of boxes in the middle layer indicate many namespace optimal write units described by NPWA (refer to Figure 97) and NPWG (refer to Figure 97), 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 97) may be placed and tracked together on the media, or the NPWG may be related to NVM subsystem data reliability implementation constraints. NPWG and NPWA attributes apply to Copy, Write, Write Uncorrectable, and Write Zeroes commands.

Figure 144: Non-conformant Write Impact



Shown in Figure 144 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 storage would consist 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 would remove the requirement to read the preceding existing data. Host writes with a length that is a multiple of NPWG would remove 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 143 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 145 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 146 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 145: Host write I/O command following NPWA and NPWG

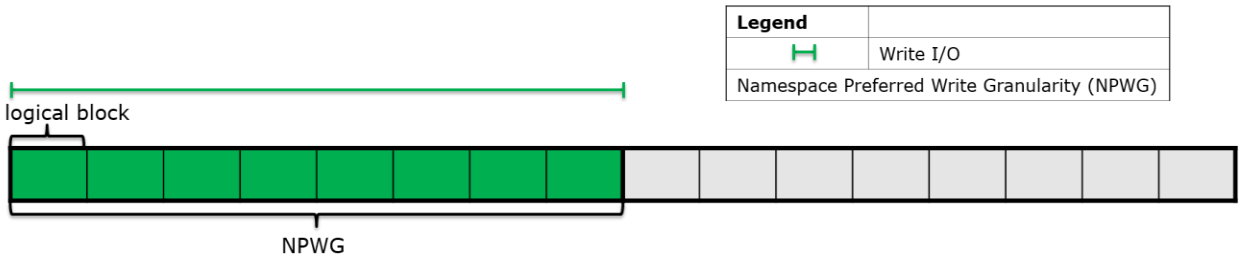
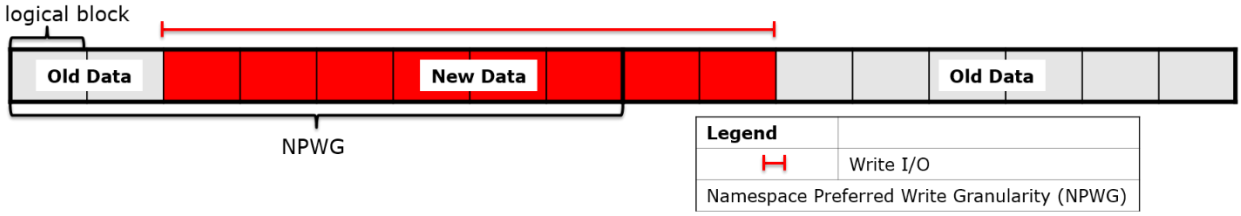
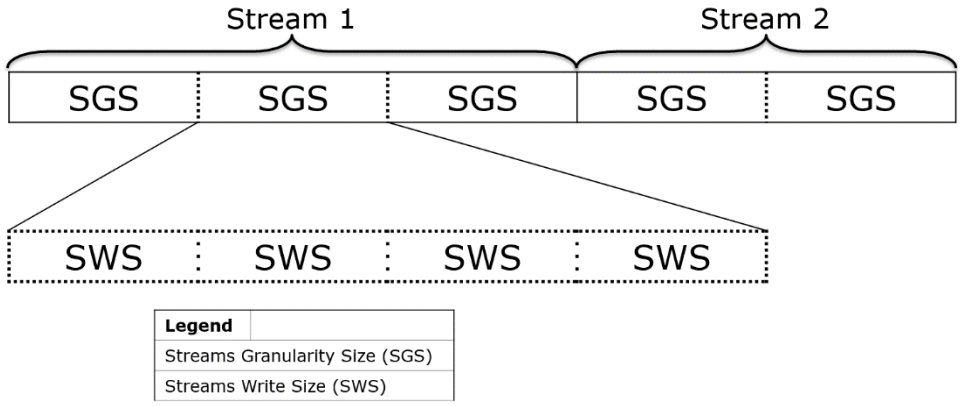


Figure 146: Host write I/O command following NPWG but not NPWA attributes



NPDG and NPDA (refer to Figure 97) are constructs in the namespace intended to improve performance for Dataset Management deallocate operations within a namespace. NPDG and NPDA may be impacted by multiple factors including but not limited to the boundaries described in Figure 143, device hardware limits, or non-volatile storage erase block sizes. Deallocating at multiples of NPDG size and aligned to NPDA ((Starting LBA modulo NPDA) == 0) may enable improved deallocate performance for the namespace.

Figure 147: Two streams composed of SGS and SWS



Streams (refer to the Streams Directive section in the NVMe Base Specification) may or may not be utilized with different namespace attributes. Figure 147 shows the streams attributes of Stream Granularity Size (SGS) and Stream Write Size (SWS) (refer to the Streams Directive – Return Parameters Data Structure figure in the NVMe Base Specification). The first stream is constructed by the host to be composed of three SGS units, and each SGS unit in this example is equal to four SWS units. The host streams are optimized for performance of the Dataset Management deallocate operations by extending the stream in units of SGS. The streams receive optimal host write performance if write I/O command lengths are integer multiples of SWS.

Streams are sometimes handled by separate I/O paths in the device. This may entail such things as different device hardware, media mapping, or reliability protections. SWS should be a multiple of the NPWG. SGS and NPDG may be equivalent to or multiples of each other. A namespace utilizing integer multiple relationships between the streams attributes (SWS and SGS) and the namespace attributes (NPWG and NPDG) may provide optimal performance by adhering to the largest attribute for write I/O commands or deallocations.

Not all namespaces describe both their Streams and namespace attributes in multiples as described above. The recommended order of priority for a host to adhere to conflicting namespace and Streams attributes is

to conform to SGS and SWS while utilizing the Streams directives. When not utilizing the Streams directives, the namespace attributes for each namespace should provide improved performance.

If the Streams Directive is enabled on a namespace, and a deallocate operations specifies logical blocks that are associated with a stream, then the host should use the SGS based alignment and size preferences in favor of the Namespace and NVM Set preferences. If the Streams Directive is not enabled on a namespace, or the logical blocks are not associated with a stream, then the host should construct deallocate operations that conform to NPDG and NPDA.

Namespace Optimal Write Size (NOWS) (refer to Figure 97) 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 allocated 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 97, 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.

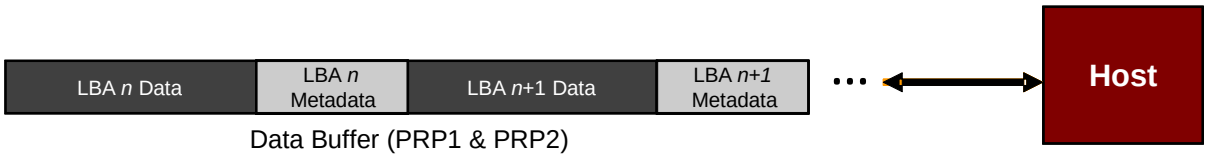
5.8.3 Metadata Handling

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.

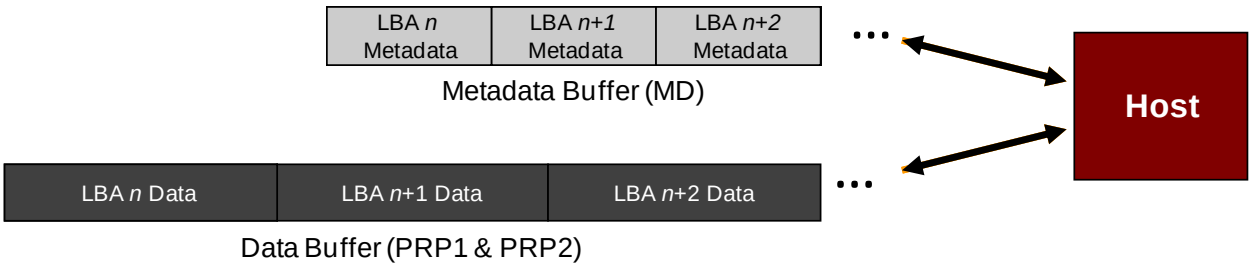
The first mechanism for transferring the metadata is as a contiguous part of the logical block that the metadata is associated with. The metadata is transferred at the end of the associated logical block, forming an extended logical block. This mechanism is illustrated in Figure 148. In this case, both 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 148: 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 149. 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 149: 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.