LEGAL NOTICE:

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

NVM Express Technical Proposal for New Feature

Technical Proposal ID	**4068b Protection Information Enhancement**
Change Date	**2021-03-35**
Builds on Specification	**NVM Express 1.4** **TP 4053a Zoned Namespaces** **TP 4065b Copy command**
References Specification	**TP 4056 Namespace Types** **NVMe 1.4 ECN 001**

Technical Proposal Author(s)

Name	**Company**
Mike Allison, Jonathan Hughes, Nick Adams, Peter Onufryk	Intel
Lee Prewitt	Microsoft
David Black, Kevin Marks	DellEMC
Martin Petersen	Oracle

This proposal defines a new 16 bytes protection information with two separate formats to provide:

- A 32b Guard field
- A 64b Guard Field

A new Storage Tag field is defined that has a viable size.

The Logical Block Reference Tag field has a variable size.

Revision History

Revision Date	**Change Description**
2020-03-18	Initial version
2020-03-19	Incorporating the following changes suggested by Peter Onufryk and David Black: • Renamed the Protection Information Format 1-3 as this is confusing. Took Peter's suggestion to put CRC size in the name as that is the differentiator. • Remove the requirement that the 32b CRC Protection and 64b CRC Protection be supported. I updated the LBA Format Data Structure to allow the LBA formats to indicate the Protection information supported per LBA format. This way the host enables the end-to-end data protection with no changes and the protection information format is known.

2020-03-23	Incorporating editorial changes from Kiel Boyle and the following: • Use “Guard” in the protection information formats instead of the CRC to match the actual field name driving the difference. • But “Check” at the end of the names for the PRINFO definition. Also aligned the command name of the DSTC in the definition of the bit. • Removed “An NVM subsystem shall support 16b Guard Protection Information.” from section 8.3. Since each LBA format defines the exact protection information format supported, then it is up to the NVM subsystem to define what it supports. • Updated the definition of the Storage Tag to indicate that it may be used to disable checking of protection information. It is not completely opaque. • Renamed Storage Tag Size from Reference Tag (STSFRT) to Derived Storage Tag from Reference Tag (DSTFRT) to match the new meaning. Updated figures FIG_DST_DRT and FIG_ST_RT to show the range of the DSTFRT instead of the boundary. • Spelled out MSB and LSB so that “B” in not confused between bytes and bits. • Updated the definition of DSTC bit to indicate that the bit is ignored when no Derived Storage Tag is defined by the configuration of the protection information format.
2020-03-24	Incorporated editorial changes from Fred Knight and the following: • Fixed the name of DSTM to be LBDSTM in Figure X2 • Clarified that PIL fixed value based on version • Changed “every” to “each” • Corrected language of DSCT field in Compare command and Write Zeroes to reflect the text in PRINFO field about PRCHK. • Removed name of polynomial name • Updated the definition of Reference Tag by the 64b Guard Protection Information to refer to the definition by the 16b Guard Protection Information since they are the same. • Added notes on new Copy command format where Key Per I/O field will go once integrated with TP 4055.
2020-03-25	Moved the selection of protection information to namespace creation and formatting. This is so that the number of LBA formats do not have to increase due to this TP as advised by Jonathan Hughes. Without the change, if a current LBA format that wanted to support all three protection information formats would add two additional LBA formats of the 16 available (13%).
2020-04-06	Moved the PI format definition to the LBA formats. Removing the “Derived” and using a merged tag field, named Logical Block Tag, that then defines multiple fields within it.
2020-04-07	Copy command used bytes in Identify Namespace data structure that overlapped the extension of the LBA formats. Therefore, moved to the new data structure form TP 4056. Discovered no mechanism for discovering on 8b PI format using Storage Tag so added a support. Added a Storage Tag Size granularity of 2^N bits where N can range between 0 and 3. Updated Namespace Granularity to support the added LBA formats.
2020-04-08	Correct the use of 16B to 16b in referring to the 16b Guard Protection Information as reported by Zhao Lian Xun.
2020-04-09	Moved STS to LBA format. Removed granularity of STS and imbedded in STS supported by controller. Renamed “Reference Tag” to “Logical Block Reference Tag”. This version approved for Phase 2 exit vote.
2020-04-10	Added the support for TP 4055 Key Per I/O to the new Descriptor Format on the Copy command.
2020-04-15	Moved LBA formats to Identify Namespace data structure as TP 4065 was able to move conflicting parameters.
2020-04-17	Jack Ellis pointed out a Logical Block Reference Tag value of FFFFFFFFh that needs to address bytes. Updated the CDW2, CWD3, and CWD14 diagram to make it clear that the bits used for LBST/ELBST and ILBRT/ELBRT is aligned to 48 bits.

2020-04-19	Added in the old diagram for 16b Guard Protection Information when STS field is cleared to 0h. Members want old diagrams due to its understanding over the history. Removed comments already covered.
2020-04-20	Aligned to recent changes to TP 4065a for the Copy command source ranges. It did not make sense to have two separate reserved sections in the Copy command source range format 1h so reduced to just one.
2020-04-21	Added the NVM Express Zoned Namespace Command Set Specification changes to increase the LBA Format Extensions.
2020-04-23	Renamed “Logical Block Tag” to “Storage and Reference Space”. Tried to move new Identify Namespace data structures from old location to new – but out reserved space. Accidently printed and caused figure numbers to be updated and had to redo all reference.
2020-05-06	As reported by Paul Suhler, the 64b CRC polynomial X^1 should have been X^0 to match Normal value.
2020-05-07	Added Paul Suhler’s 64-bit CRC definition.
2020-05-15	Filled in byte count of 64b test case. Formatting. Changed 64-bit to 64b. Changed “*” to “x” in formulas. Updates N to be 4 KiB in the 64b test cases. Moved the definition of the value 0h in the definition of the Storage Tag Size field until after the min and max values have been described per response to Yoni Shternhell.
2020-05-20	Corrected the STS value in Figure FIG_STS for the 32b Protection Information as identified by Jackson Ellis.
2020-05-21	Corrected Figure 246 64b STS maximum value to be 32 as identified by Ee Loon Teoh.
2020-05-26	Filled in the 64b CRC test cases.
2020-05-29	Added the Rocksoft™ Model CRC Algorithm parameters for the 64b CRC.
2020-06-04	Corrected misspelling. Formatting. Made sure all references are highlighted. Made sure TP 4055 text specifies that the text needs to be red in the integrated document. Aligned with member review copy of TP 4056 for section 3.1.1 of TP 4053. Added End-to-end Storage Tag Check Error to list of check errors.
2020-06-17	Updates FLBAS field and Format NVM to allow 64 LBA formats.
2020-06-18	The new fields to FLBAS field and Format NVM to allow 64 LBA formats are not compatible is a host that supports this TP interacts with a controller that does not. Added text to describe that the host must set the new fields to zero to avoid undefined behavior.
2020-07-08	Added using the Host Behavior Feature to allow the host to indicate when this TP is supported. If not supported, then the controller is not allowed to report allocated namespace information to the host if those namespaces are formatted with the new protection information capabilities. This is to allow the device to work with a host that does not support this TP.
2020-07-09	Limited the values in the new Host Behavior item to 0 and 1.
2020-07-14	The STS field in the LBA Format data structure was defined as a bit and needed to be a byte. When extended to a byte, the new fields defined for this TP did not fit into the LBA Format data structure. Added text to I/O commands, Format NVM command, and Namespace Management command indicating that if the host does not support the new Protection Information formats, then the commands are aborted.
2020-07-15	There were two instances of the NVM Command Set Identify Namespace Data Structure so I combined the information.
2020-07-16	Compressed the data in the Extended LBA Format to use only the bits required. Move the controller support for the new protection formats to the Identify Controller data structure. Editorial edits on the Host Behavior feature additions. Updated Figure X2_A to make it clear the byte (383:0) that are a subset of the Identify Namespace data structure and the byte (384 and greater) that are not associated with the Identify Namespace data structure.
2020-07-18	Made all new fields in the Figure X2 NVM Command Set Identify Namespace Data Structure optional.

2020-07-21	Made sure figure and section reference are highlighted. Formatting changes.
2020-07-23	Editorial changes. Added the Zone Append command defined by TP 4053.
2020-07-31	Fixed the bits assignments to Zone Append command as pointed out by Paul Suhler.
2020-08-04	In Phase 2 exit a request was made to maximize Storage Tag sizes up to a 64-bit field instead on guaranteeing a Reference Tag size of 16-bits to allow maximum flexibility.
2020-08-04	Storage Tag Size changed to 7 bits as reported by Jiafen Yuan. Based on Jiafen's comment a review of the bits / bytes cause the following updates: • Fixed the reserved byte offsets in Figure X2 • Fixed the reserved bit offsets in Figure 238 • Changes Command Dword 3 to Command Dword 14 in Figure FIG_CMD_SDWS • No need to have TBD in the byte offsets for LBA Format Extension • Added the section number for ZNS changes • Removed ZNS sections that have not changed.
2020-08-05	Added reference to status code when Format NVM and Namespace Management. Editorial change to the 16BPISTS field. Fixed the ELBA bit to reflect that when set to '1' the 16b Guard Protection information supports a STS field cleared to 0h. Modified item 2 of the LBAFEE field with editorial suggestions and added that the actual protection information formats are defined by the extended LBA formats. Made sure Storage Tag and Logical Block Reference Tag are referenced to as fields. Added "(STS)" when Storage Tag Size field is mentioned. Updated section 8.3.TBD.4 to reflect that the Logical Block Reference Tag is allowed to not be defined.
2020-08-17	Phase 2 exit vote passed. Making the file
2020-08-20	Technical WG approved member review. Removed comments and accepted all changes.
2020-09-24	Removed the reference to Storage Tag value causing protection information checking to be disabled. Ready for integration.
2020-10-07	Integrated into the NVM Express Base Specification.
2020-10-08	Fixed alignment of text in cells. Ready for ratification ready review.
2020-10-15	Removed all comments and accepted changes for ratification.
2020-10-23	Fixing the byte references to Reference Tag as that field is a bit sized field. The Reference Tag may not be defined. If not defined, then turning off checking should be by the Application Tag only.
2020-10-30	Editorial changes on the text for the Reference Tag disabling protection information checking. Added the roll over definition for the computer reference tag.
2020-11-03	Corrected the CTRATT field as the new bit should have been bit 15.
2020-11-03	Renamed to TP 4068b as a different TP 4068a is needed to fix an integration issue. Applied TP 4068a changes.
2020-11-03a	Added CRC-32C test cases, using CRC values from Mike Allison.
2020-12-02	Added the Directives added to the Append command added to TP 5053a
2020-12-17	Accepted all changes and removed all comments for member review.
2021-01-28	Changed lists from bullets to lower case letters if list was defined by this proposal. Removed a sentence with two if's by creating two sentences. Removed showing LBA Format 62 Support.
2021-01-31	Integrated into the NVMe Base Specification.
2021-02-04	Added a suggested status code on aborting a Copy command due to the Descriptor Format field value not matching the specified namespace format.
2021-02-13	Clarifying the definition of the variable ELBST and EILBRT fields in commands.
2021-02-16	Correct pluralization by changing "specifies" to "specify". Changed Dowrd to Dword. Changed a reference to TP 4056 to green per the convention of the TP.

3/5/2021	Updated the Host Behavior to clearly state the field is for the host to indicate support. Updated the Optional Copy Formats Supported field defined by TP 4065b. Aligned with TP 4065b.
3/11/2021	Accepted all changes, removed all comments, and converted references/cross-references to text.
3/21/2021	Integrated into the NVMe Base Specification
3/25/2021	Accepted all changes, removed all comments, and converted references/cross-references to text for ratification.

Description for NVMe 1.4 Changes Document

This technical proposal adds:

- Two new protection information formats.
- Read and write commands use CDW2, CDW3, and CDW14 two transfer the variable sized information to for the Storage Tag field and Logical Block Reference Tag field.
- Added a new descriptor format to support the new protection information formats.
- Allows the existing 8B protection information format use a Storage Tag field from the Logical Block Reference Tag field.

This technical proposal removed support for protection information being in the first bytes of the metadata for versions later than version 1.4 of this specification and is incompatible with previous revisions.

Description of Specification Changes

Markup Conventions:

Black:	Unchanged (however, hot links are removed)
~~Red Strikethrough~~:	Deleted
Blue:	New
Blue Highlighted:	TBD values, anchors, and links to be inserted in new text.
<Green Bracketed>:	Notes to editor

Modify portions of NVMe 1.4 as shown below:

Modify portions of Figure 105 in section 4.2 as shown below <note this is going into TP 4056 but added here for clarity during this TP review for Phase 2>:

4.2 Submission Queue Entry – Command Format

...

Figure 105: Command Format – Admin and NVM Command Set

Bytes	Description
03:00	**Command Dword 0 (CDW0):** This field is common to all commands and is defined in Figure 104.

Figure 105: Command Format – Admin and NVM Command Set

Bytes	Description
07:04	**Namespace Identifier (NSID):** This field specifies the namespace that this command applies to. If the namespace identifier is not used for the command, then this field shall be cleared to 0h. The value FFFFFFFFh in this field is a broadcast value (refer to section 6.1), where the scope (e.g., the NVM subsystem, all attached namespaces, or all namespaces in the NVM subsystem) is dependent on the command. Refer to Figure 139, Figure 140 and Figure 346 for commands that support the use of the value FFFFFFFFh in this field. Specifying an inactive namespace identifier (refer to section 6.1.4) in a command that uses the namespace identifier shall cause the controller to abort the command with status Invalid Field in Command, unless otherwise specified. Specifying an invalid namespace identifier (refer to section 6.1.2) in a command that uses the namespace identifier shall cause the controller to abort the command with status Invalid Namespace or Format, unless otherwise specified. If the namespace identifier is used for the command (refer to Figure 139 and Figure 140), the value FFFFFFFFh is not supported for that command, and the host specifies a value of FFFFFFFFh, then the controller should abort the command with status Invalid Field in Command, unless otherwise specified. If the namespace identifier is not used for the command and the host specifies a value from 1h to FFFFFFFEh, then the controller should abort the command with status Invalid Field in Command, unless otherwise specified.
11:08	~~Reserved~~ **Command Dword 2 (CDW2):** This field is command specific Dword 2.
15:12	**Command Dword 3 (CDW3):** This field is command specific Dword 3.
23:16	**Metadata Pointer (MPTR):** This field is valid only if the command has metadata that is not interleaved with the logical block data, as specified in the Format NVM command. This is a reserved field in NVMe over Fabrics implementations. If CDW0.PSDT (refer to Figure 104 is cleared to 00b, then this field shall contain the address of a contiguous physical buffer of metadata and that address shall be dword aligned (i.e., bits 1:0 cleared to 00b). The controller is not required to check that bits 1:0 are cleared to 00b. The controller may report an error of Invalid Field in Command if bits 1:0 are not cleared to 00b. If the controller does not report an error of Invalid Field in Command, then the controller shall operate as if bits 1:0 are cleared to 00b. If CDW0.PSDT is set to 01b, then this field shall contain the address of a contiguous physical buffer of metadata and that address may be aligned on any byte boundary. If CDW0.PSDT is set to 10b, then this field shall contain the address of an SGL segment that contains exactly one SGL Descriptor. The address of that SGL segment shall be qword aligned (i.e., bits 2:0 cleared to 000b). The SGL Descriptor contained in that SGL segment is the first SGL Descriptor of the metadata for the command. If the SGL Descriptor contained in that SGL segment is an SGL Data Block descriptor, then that SGL Data Block Descriptor is the only SGL Descriptor and therefore describes the entire metadata data transfer. Refer to section 4.4. The controller is not required to check that bits 2:0 are cleared to 000b. The controller may report an error of Invalid Field in Command if bits 2:0 are not cleared to 000b. If the controller does not report an error of Invalid Field in Command, then the controller shall operate as if bits 2:0 are cleared to 000b.

Figure 105: Command Format – Admin and NVM Command Set

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>39:24</td><td>
Data Pointer (DPTR): This field specifies the data used in the command.

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

<table>
<tr><td>39:32</td><td>PRP Entry 2 (PRP2): This field:
a) is reserved if the data transfer does not cross a memory page boundary;
b) specifies the Page Base Address of the second memory page if the data transfer crosses exactly one memory page boundary. E.g.,:
i. the command data transfer length is equal in size to one memory page and the offset portion of the PBAO field of PRP1 is non-zero; or
ii. the Offset portion of the PBAO field of PRP1 is equal to 0h and the command data transfer length is greater than one memory page and less than or equal to two memory pages in size;
and
c) is a PRP List pointer if the data transfer crosses more than one memory page boundary. E.g.,:
i. the command data transfer length is greater than or equal to two memory pages in size but the offset portion of the PBAO field of PRP1 is non-zero; or
ii. the command data transfer length is equal in size to more than two memory pages and the Offset portion of the PBAO field of PRP1 is equal to 0h.</td></tr>
<tr><td>31:24</td><td>PRP Entry 1 (PRP1): This field contains the first PRP entry for the command or a PRP List pointer depending on the command.</td></tr>
</table>

If CDW0.PSDT is set to 01b or 10b, then the definition of this field is:

<table>
<tr><td>39:24</td><td>SGL Entry 1 (SGL1): This field contains the first SGL segment for the command. If the SGL segment is an SGL Data Block or Keyed SGL Data Block or Transport SGL Data Block descriptor, then it describes the entire data transfer. If more than one SGL segment is needed to describe the data transfer, then the first SGL segment is a Segment, or Last Segment descriptor. Refer to section 4.4 for the definition of SGL segments and descriptor types.

The NVMe Transport may support a subset of SGL Descriptor types and features as defined in the NVMe Transport binding specification.</td></tr>
</table>
</td></tr>
<tr><td>43:40</td><td>Command Dword 10 (CDW10): This field is command specific Dword 10.</td></tr>
<tr><td>47:44</td><td>Command Dword 11 (CDW11): This field is command specific Dword 11.</td></tr>
<tr><td>51:48</td><td>Command Dword 12 (CDW12): This field is command specific Dword 12.</td></tr>
<tr><td>55:52</td><td>Command Dword 13 (CDW13): This field is command specific Dword 13.</td></tr>
<tr><td>59:56</td><td>Command Dword 14 (CDW14): This field is command specific Dword 14.</td></tr>
<tr><td>63:60</td><td>Command Dword 15 (CDW15): This field is command specific Dword 15.</td></tr>
</table>

In addition to the fields commonly defined for all Admin and NVM commands, Admin and NVM Vendor Specific commands may support the Number of Dwords in Data Transfer and Number of Dwords in Metadata Transfer fields. If supported, the command format for the Admin Vendor Specific Command and NVM Vendor Specific Commands are defined in Figure 106. For more details, refer to section 8.7.

Figure 106: Command Format – Admin and NVM Vendor Specific Commands (Optional)

Bytes	Description
03:00	**Command Dword 0 (CDW0):** This field is common to all commands and is defined in Figure 104.

Figure 106: Command Format – Admin and NVM Vendor Specific Commands (Optional)

Bytes	Description
07:04	**Namespace Identifier (NSID):** This field indicates the namespace ID that this command applies to. If the namespace ID is not used for the command, then this field shall be cleared to 0h. Setting this value to FFFFFFFFh causes the command to be applied to all namespaces attached to this controller, unless otherwise specified. The behavior of a controller in response to an inactive namespace ID for a vendor specific command is vendor specific. Specifying an invalid namespace ID in a command that uses the namespace ID shall cause the controller to abort the command with status Invalid Namespace or Format, unless otherwise specified.
15:08	Reserved
39:16	Refer to Figure 105 for the definition of these fields.
43:40	**Number of Dwords in Data Transfer (NDT):** This field indicates the number of dwords in the data transfer.
47:44	**Number of Dwords in Metadata Transfer (NDM):** This field indicates the number of dwords in the metadata transfer.
51:48	**Command Dword 12 (CDW12):** This field is command specific Dword 12.
55:52	**Command Dword 13 (CDW13):** This field is command specific Dword 13.
59:56	**Command Dword 14 (CDW14):** This field is command specific Dword 14.
63:60	**Command Dword 15 (CDW15):** This field is command specific Dword 15.

Modify portions of section 4.5 as shown below:

4.5 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) or metadata may be transferred as a separate buffer of data. The metadata shall not be split between the logical block and a separate metadata buffer. For writes, the metadata shall be written atomically with its associated logical block. Refer to section 8.2.

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

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

If the namespace is formatted to use end-to-end data protection (refer to section 8.3), then the first ~~eight~~ bytes or last ~~eight~~ bytes of the metadata is used for protection information (specified as part of the NVM Format operation).

Modify portions of Figure 131 in section 4.6.1.2.3 as shown below:

4.6.1.2.3 Media and Data Integrity Errors Definition

...

Figure 131: Status Code – Media and Data Integrity Error Values, NVM Command Set

Value	Description
80h	**Write Fault:** The write data could not be committed to the media.
81h	**Unrecovered Read Error:** The read data could not be recovered from the media.
82h	**End-to-end Guard Check Error:** The command was aborted due to an end-to-end guard check failure.

Figure 131: Status Code – Media and Data Integrity Error Values, NVM Command Set

Value	Description
83h	**End-to-end Application Tag Check Error:** The command was aborted due to an end-to-end application tag check failure.
84h	**End-to-end Reference Tag Check Error:** The command was aborted due to an end-to-end logical block reference tag check failure.
85h	**Compare Failure:** The command failed due to a miscompare during a Compare command.
86h	**Access Denied:** Access to the namespace and/or LBA range is denied due to lack of access rights. Refer to the appropriate security specification (e.g., TCG Storage Interface Interactions specification).
87h	**Deallocated or Unwritten Logical Block:** The command failed due to an attempt to read from or verify an LBA range containing a deallocated or unwritten logical block.
88h	**End-to-end Storage Tag Check Error:** The command was aborted due to an end-to-end storage tag check failure.
89h to BFh	Reserved

Modify portions of Figure 245 and Figure 246 in section 5.15.2.1 as shown below:

5.15.2.1 Identify Namespace data structure (CNS 00h)

...

Figure 245: Identify – Identify Namespace Data Structure, NVM Command Set Specific

Bytes	O/M 1	Description
...		
25	M	**Number of LBA Formats (NLBAF):** This field defines the number of supported LBA data size and metadata size combinations supported by the namespace. LBA formats shall be allocated in order (starting with 0) and packed sequentially. This is a 0's based value. The maximum number of LBA formats that may be indicated as supported is: a) 16 if the host has cleared the LBA Format Extension Enable (LBAFEE) field to 0h in the Host Behavior Support feature (refer to section 5.21.1.22); or b) 64 if the host has set the LBAFEE field to 1h in the Host Behavior Support feature (refer to section 5.21.1.22). 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 8.2.

Figure 245: Identify – Identify Namespace Data Structure, NVM Command Set Specific

Bytes	O/M 1	Description
26	M	**Formatted LBA Size (FLBAS):** This field indicates the LBA data size & metadata size combination that the namespace has been formatted with (refer to section 5.23). Bit 7 is reserved. Bits 6:5 indicate the most significant 2 bits of the supported LBA Formats indicated in this data structure. These bits are ignored if the number of supported LBA Formats is less than or equal to 16. Bit 4 if set to '1' indicates that the metadata is transferred at the end of the data LBA, creating an extended data LBA. Bit 4 if cleared to '0' indicates that all of the metadata for a command is transferred as a separate contiguous buffer of data. Bit 4 is not applicable when there is no metadata. Bits 3:0 indicates the least significant 4 bits of one of the ~~16~~ supported LBA Formats indicated in this data structure.

Figure 245: Identify – Identify Namespace Data Structure, NVM Command Set Specific

<table>
<tr><th>Bytes</th><th>O/M
1</th><th>Description</th></tr>
<tr><td>28</td><td>M</td><td>End-to-end Data Protection Capabilities (DPC): This field indicates the capabilities for the end-to-end data protection feature. Multiple bits may be set in this field. Refer to section 8.3.
~~Bits 7: 5 are reserved.~~
~~Bit 4 if set to '1' indicates that the namespace supports protection information transferred as the last eight bytes of metadata. Bit 4 if cleared to '0' indicates that the namespace does not support protection information transferred as the last eight bytes of metadata.~~
~~Bit 3 if set to '1' indicates that the namespace supports protection information transferred as the first eight bytes of metadata. Bit 3 if cleared to '0' indicates that the namespace does not support protection information transferred as the first eight bytes of metadata.~~
~~Bit 2 if set to '1' indicates that the namespace supports Protection Information Type 3. Bit 2 if cleared to '0' indicates that the namespace does not support Protection Information Type 3.~~
~~Bit 1 if set to '1' indicates that the namespace supports Protection Information Type 2. Bit 1 if cleared to '0' indicates that the namespace does not support Protection Information Type 2.~~
~~Bit 0 if set to '1' indicates that the namespace supports Protection Information Type 1. Bit 0 if cleared to '0' indicates that the namespace does not support Protection Information Type 1.~~
<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 ~~eight~~ bytes of metadata. ~~Bit 4 i~~If cleared to '0' indicates that the namespace does not support protection information transferred as the last ~~eight~~ 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 ~~eight~~ bytes of metadata. ~~Bit 3 i~~If cleared to '0' indicates that the namespace does not support protection information transferred as the first ~~eight~~ bytes of metadata. For versions later than version 1.4 of this 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. ~~Bit 2 i~~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. ~~Bit 1 i~~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. ~~Bit 0 i~~If cleared to '0' indicates that the namespace does not support Protection Information Type 1.</td></tr>
</table>
</td></tr>
</table>

Figure 245: Identify – Identify Namespace Data Structure, NVM Command Set Specific

<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 8.3.
~~Bits 7:4 are reserved.~~
~~Bit 3 if set to '1' indicates that the protection information, if enabled, is transferred as the first eight bytes of metadata. Bit 3 if cleared to '0' indicates that the protection information, if enabled, is transferred as the last eight bytes of metadata.~~
~~Bits 2:0 indicate whether Protection Information is enabled and the type of Protection Information enabled. The values for this field have the following meanings:~~
<table>
<tr><th>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 ~~eight~~ bytes of metadata. Bit 3 if cleared to '0' indicates that the protection information, if enabled, is transferred as the last ~~eight~~ bytes of metadata. For versions later than version 1.4 of this specification this bit shall be cleared to '0'.</td></tr>
<tr><td>2:0</td><td>Protection Information Type (PIT): This field indicates whether p~~P~~rotection i~~I~~nformation is enabled and the type of p~~P~~rotection i~~I~~nformation 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>…</td><td></td><td></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 ~~FF~~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>
<tr><td>…</td><td></td><td></td></tr>
</table>

Figure 245: Identify – Identify Namespace Data Structure, NVM Command Set Specific

Bytes	O/M 1	Description
131:128	O	**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 246.
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 246**.**
~~139:136~~	~~O~~	~~**LBA Format 2 Support (LBAF2):** This field indicates the LBA format 2 that is supported by the controller. The LBA format field is defined in Figure 246**.**~~
~~143:140~~	~~O~~	~~**LBA Format 3 Support (LBAF3):** This field indicates the LBA format 3 that is supported by the controller. The LBA format field is defined in Figure 246**.**~~
~~147:144~~	~~O~~	~~**LBA Format 4 Support (LBAF4):** This field indicates the LBA format 4 that is supported by the controller. The LBA format field is defined in Figure 246.~~
~~151:148~~	~~O~~	~~**LBA Format 5 Support (LBAF5):** This field indicates the LBA format 5 that is supported by the controller. The LBA format field is defined in Figure 246.~~
~~155:152~~	~~O~~	~~**LBA Format 6 Support (LBAF6):** This field indicates the LBA format 6 that is supported by the controller. The LBA format field is defined in Figure 246.~~
~~159:156~~	~~O~~	~~**LBA Format 7 Support (LBAF7):** This field indicates the LBA format 7 that is supported by the controller. The LBA format field is defined in Figure 246.~~
~~163:160~~	~~O~~	~~**LBA Format 8 Support (LBAF8):** This field indicates the LBA format 8 that is supported by the controller. The LBA format field is defined in Figure 246.~~
~~167:164~~	~~O~~	~~**LBA Format 9 Support (LBAF9):** This field indicates the LBA format 9 that is supported by the controller. The LBA format field is defined in Figure 246.~~
~~171:168~~	~~O~~	~~**LBA Format 10 Support (LBAF10):** This field indicates the LBA format 10 that is supported by the controller. The LBA format field is defined in Figure 246.~~
~~175:172~~	~~O~~	~~**LBA Format 11 Support (LBAF11):** This field indicates the LBA format 11 that is supported by the controller. The LBA format field is defined in Figure 246.~~
~~179:176~~	~~O~~	~~**LBA Format 12 Support (LBAF11):** This field indicates the LBA format 12 that is supported by the controller. The LBA format field is defined in Figure 246.~~
~~183:180~~	~~O~~	~~**LBA Format 13 Support (LBAF13):** This field indicates the LBA format 13 that is supported by the controller. The LBA format field is defined in Figure 246.~~
~~187:184~~	~~O~~	~~**LBA Format 14 Support (LBAF14):** This field indicates the LBA format 14 that is supported by the controller. The LBA format field is defined in Figure 246.~~
~~191:188~~	~~O~~	~~**LBA Format 15 Support (LBAF14):** This field indicates the LBA format 15 that is supported by the controller. The LBA format field is defined in Figure 246.~~
…		
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 246.

The LBA format data structure is described in Figure 246.

Figure 246: Identify – LBA Format Data Structure, NVM Command Set Specific

Bits	Description
31:26	Reserved

Figure 246: Identify – LBA Format Data Structure, NVM Command Set Specific

<table>
<tr><th>Bits</th><th>Description</th></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>

Modify Figure X2 from TP 4056 in section 5.15.2.TBDa.1 (NVM Command Set Identify Namespace Data structure (CSI 00h)) shown below:

5.15.2.TBDa.1 NVM Command Set Identify Namespace Data structure (CSI 00h)

…

Figure X2: NVM Command Set Identify Namespace Data Structure

Bytes	O/M 1	Description
7:0	O	**Logical Block Storage Tag Mask (LBSTM):** Identifies the mask for the Storage Tag field for the protection information (refer to section 8.3). The size of this field is define by the STS field. If the size of this field is less than 64 bits, the mask is contained in the least significant bits of this field. If end-to-end protection is not enabled in the namespace, then this field is ignored. 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', then all bits in the mask shall be set to '1'.

Figure X2: NVM Command Set Identify Namespace Data Structure

<table>
<tr><th>Bytes</th><th>O/M 1</th><th>Description</th></tr>
<tr><td>8</td><td>O</td><td>Protection Information Capabilities (PIC): This field indicates the capabilities for the protection information formats.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:2</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 8.3.TBD.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 245 and Figure X2), 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>15:12</td><td>O</td><td>Extended LBA Format 0 Support (ELBAF0): This field indicates additional LBA format information to the LBA Format 0 Support (ELBAF0) field in the Identify Namespace data structure. The Extended LBA format field is defined in Figure FIG_ELBAF.</td></tr>
<tr><td>19:16</td><td>O</td><td>Extended LBA Format 1 Support (ELBAF1): This field indicates additional LBA format information to the LBA Format 1 Support (ELBAF1) field in the Identify Namespace data structure. The Extended LBA format field is defined in Figure FIG_ELBAF.</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 information to the LBA Format 63 Support (ELBAF63) field in the Identify Namespace data structure. The Extended LBA format field is defined in Figure FIG_ELBAF.</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 FIG_ELBAF.

Figure FIG_ELBAF: Identify – 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 8.3.TBD) 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>
</table>

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

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>6:0</td><td>Storage Tag Size (STS): Indentifies the number of most significant bits of the protection information Storage and Reference Space field that define the Storage Tag field (refer to section 8.3.TBD.4).

This field does limit the minimum and maximum values allowed per protection information formats (refer to section 8.3.TBD):

<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>

Modify portions of Figure 247 and in section 5.15.2.2 as shown below:

5.15.2.2 Identify Controller data structure (CNS 01h)

...

Figure 247: Identify – Identify Controller Data Structure

Bytes	O/M [1]	Description
...		

<table>
<tr>
<td>99:96</td>
<td>M</td>
<td>Controller Attributes (CTRATT): This field indicates attributes of the controller.
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:16</td><td>~~are r~~Reserved~~.~~</td></tr>
<tr><td>15</td><td>Extended LBA Formats Supported (ELBAS): If set to ‘1’ indicates that the controller supports the protection information formats (refer to section 8.3.TBD):
a) 16b Guard Protection Information with no additional restrictions on the STS field other than defined by the STS field (refer to Figure FIG_ELBAF);
b) 32b Guard Protection Information; and
c) 64b Guard Protection Information.

If cleared to ‘0’ indicates that the controller does not support the protection information formats (refer to section 8.3.TBD):
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.

Refer to the LBA Format Extension Enable (LBAFEE) field in the Host Behavior Support feature (refer to section 5.21.1.22) for details for host software to enable the controller to operate on namespaces using the protection information formats.</td></tr>
<tr><td>9</td><td>~~Bit 9 (~~UUID List~~):~~ If set to ‘1’, then the controller supports reporting of a UUID List (refer to Figure 257). If cleared to ‘0’, then the controller does not support reporting of a UUID List (refer to section 8.24).</td></tr>
<tr><td>8</td><td>~~Bit 8 (~~SQ Associations~~):~~ If set to ‘1’, then the controller supports SQ Associations (refer to section 8.23). If cleared to ‘0’, then the controller does not support SQ Associations.</td></tr>
<tr><td>7</td><td>~~Bit 7 (~~Namespace Granularity~~):~~ If set to ‘1’, then the controller supports reporting of Namespace Granularity (refer to section 5.15.2.12). If cleared to ‘0’, the controller does not support reporting of Namespace Granularity. If the Namespace Management capability (refer to section 8.12) is not supported, then this bit shall be cleared to ‘0’.</td></tr>
<tr><td>6</td><td>~~Bit 6 (~~Traffic Based Keep Alive Support~~ – (~~TBKAS): If set to '1‘, then the controller supports restarting the Keep Alive Timer if an Admin command or an I/O command is processed during the Keep Alive Timeout Interval (refer to section 7.12.2). If cleared to '0‘, then the controller supports restarting the Keep Alive Timer only if a Keep Alive command is processed during the Keep Alive Timeout Interval (refer to section 7.12.1).</td></tr>
<tr><td>5</td><td>~~Bit 5 (~~Predictable Latency Mode~~):~~ If set to ‘1’, then the controller supports Predictable Latency Mode (refer to section 8.18). If cleared to ‘0’, then the controller does not support Predictable Latency Mode.</td></tr>
<tr><td>4</td><td>~~Bit 4 (~~Endurance Groups~~):~~ If set to ‘1’, then the controller supports Endurance Groups (refer to section 8.17). If cleared to ‘0’, then the controller does not support Endurance Groups.</td></tr>
<tr><td>3</td><td>~~Bit 3 (~~Read Recovery Levels~~):~~ If set to ‘1’, then the controller supports Read Recovery Levels (refer to section 8.16). If cleared to ‘0’, then the controller does not support Read Recovery Levels.</td></tr>
<tr><td>2</td><td>~~Bit 2 (~~NVM Sets~~):~~ If set to ‘1’, then the controller supports NVM Sets (refer to section 4.9). If cleared to ‘0’, then the controller does not support NVM Sets.</td></tr>
<tr><td>1</td><td>~~Bit 1 (~~Non-Operational Power State Permissive Mode~~):~~ If set to ‘1’, then the controller supports host control of whether the controller may temporarily exceed the power of a non-operational power state for the purpose of executing controller initiated background operations in a non-operational power state (i.e., Non-Operational Power State Permissive Mode supported). If cleared to ‘0’, then the controller does not support host</td></tr>
</table>
</td>
</tr>
</table>

Figure 247: Identify – Identify Controller Data Structure

Bytes	O/M [1]	Description
		control of whether the controller may exceed the power of a non-operational state for the purpose of executing controller initiated background operations in a non-operational state (i.e., Non-Operational Power State Permissive Mode not supported). Refer to section 5.21.1.17.
		0 ~~Bit 0~~ **Host Identifier Support:** ~~I~~if set to '1', then the controller supports a 128-bit Host Identifier. Bit 0 if cleared to '0', then the controller does not support a 128-bit Host Identifier.
…		
<The following is from TP 4065b>		
535:534	O	**Optional Copy Formats Supported:** ~~Bits 15:1 are reserved.~~ ~~Bit 0 if set to '1', then the controller supports Copy Format 0h. If cleared to '0', then the controller does not support Copy Format 0h.~~ **Bits** / **Description** 15:2 Reserved 1 If set to '1', then the controller supports Copy Format 1h. If cleared to '0', then the controller does not support Copy Format 1h. 0 If set to '1', then the controller supports Copy Format 0h. If cleared to '0', then the controller does not support Copy Format 0h.

Modify portions of Figure 255 and in section 5.15.2.12 as shown below:

5.15.2.12 Namespace Granularity List (CNS 16h)

If the controller supports reporting of Namespace Granularity (refer to section 8.12.1), then a Namespace Granularity List (refer to Figure 255) is returned to the host for up to:

a) 16 ~~sixteen~~ namespace granularity descriptors (refer to Figure 256) if the host has cleared the LBA Format Extension Enable (LBAFEE) field to 0h in the Host Behavior Support feature (refer to section 5.21.1.22); or
b) 64 namespace granularity descriptors if the host has set the LBAFEE field to 1h in the Host Behavior Support feature.

Figure 255: 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 245). 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.

Figure 255: Namespace Granularity List

Bytes	Description
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~~287~~:~~272~~1040	**NG Descriptor 63~~15~~:** This field contains the sixty-fourth namespace granularity descriptor in the list. This namespace granularity descriptor applies to LBA Format 63~~15~~.

The format of the namespace granularity descriptor is defined in Figure 256.

Figure 256: 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.

Modify a portion of section 5.20 as follows:

5.20 Namespace Management command

...

The data structure used for the create operation is defined in Figure 265 and has the same format as the Identify Namespace data structure defined in Figure 245. After successful completion of a Namespace Management command with the create operation, the namespace is formatted with the specified attributes. The fields that host software may specify in the create operation are defined in Figure 262. Fields that are reserved are cleared to 0h by host software. There is no data structure transferred for the delete operation.

If a host does not set the LBA Format Extension Enable (LBAFEE) field to 1h in the Host Behavior Support feature (refer to section 5.21.1.22), 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 8.3.TBD):

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.

Modify a portion of the new NVM Command Set structure for Namespace Management defined in Section 5.20.x.1 of TP 4056 (Namespace Types) as follows:

5.20.x.1 NVM Command Set (Command Set Identifier 00h)

The data structure passed to the create operation is defined in Figure X2_A. Fields that are reserved should be cleared to 0h by host software. ~~This structure is a subset of the Identify Namespace structure specified for the NVM Command Set.~~ After successful completion of a Namespace Management command with the create operation, the namespace is formatted with the specified attributes.

Figure X2_A: Namespace Management – Host Software Specified Fields

Bytes	Description	Host Specified
Fields that are a subset of the Identify Namespace data structure (refer to Figure 245)		
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)	Yes
383:102	Reserved	
Fields that are not a subset of the Identify Namespace data structure.		
391:384	Logical Block Storage Tag Mask (LBSTM)	Yes

Notes:
1. A value of 0h specifies that the controller determines the value to use (refer to section 8.12).

Modify portions of Figure 313 in section 5.21.1.22 as shown below:

5.21.1.22 Host Behavior Support (Feature Identifier 16h)

…

Figure 313: Host Behavior Support – Data Structure

Bytes	Description
00	**Advanced Command Retry Enable (ACRE):** If set to 1h, then the Command Interrupted status code is enabled (refer to Figure 126) and command retry delays are enabled. The controller may use the Command Interrupted status code and may indicate a command retry delay by setting the Command Retry Delay (CRD) field to a non-zero value in the Status field of a Completion Queue Entry, refer to Figure 124. A host that sets this field to 1h indicates host support for the command retry behaviors that are specified for both the Command Interrupted status code and non-zero values in the CRD field. If cleared to 0h, then both the Command Interrupted status code and command retry delays are disabled. The controller shall not use the Command Interrupted status code~~,~~ and shall clear the CRD field to 0h in all CQEs. All values other than 0h and 1h are reserved.

Figure 313: 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 Figure 247). 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 245 and I/O Command Set Specific Identify Namespace Data Structure in the Zoned Namespaces Command Set); and b) the number of namespace granularity descriptors (refer to Figure 255); and 2) is enabled to create, format, and perform I/O commands on namespaces formatted with (refer to section 8.3.TBD): 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 X2) 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 8.3.TBD): 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.
511:03	Reserved

Modify portions of section 5.23 as shown below:

5.23 Format NVM command – NVM Command Set Specific

...

For a Format command with an NSID field set to FFFFFFFFh that does not specify a secure erase:

a) if bit 0 is set to '1' in the FNA field and there are no namespaces in the NVM subsystem, then that Format command shall complete without error; and
b) if bit 0 is cleared to '0' in the FNA field and there are no attached namespaces, then that Format command shall complete without error.

If a host does not set the LBA Format Extension Enable (LBAFEE) field to 1h in the Host Behavior Support feature (refer to section 5.21.1.22), then a controller aborts a Format NVM command with a status code of Invalid Namespace or Format that specifies a format of (refer to section 8.3.TBD):

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

c) 64b Guard Protection Information.

…

Figure 238: Format NVM – Command Dword 10

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>31:14</td><td>Reserved</td></tr>
<tr><td>13:12</td><td>LBA Format Upper (LBAFU): This field specifies the most significant 2 bits of the LBA format to apply to the NVM media. This corresponds to the LBA formats indicated in the Identify command, refer to Figure 245 and Figure 246. Only supported LBA formats shall be selected.

This field is ignored If the host has cleared the LBA Format Extension Enable (LBAFEE) field to 0h in the Host Behavior Support feature (refer to section 5.21.1.22).</td></tr>
<tr><td>11:09</td><td>Secure Erase Settings (SES): This field specifies whether a secure erase should be performed as part of the format and the type of the secure erase operation. The erase applies to all user data, regardless of location (e.g., within an exposed LBA, within a cache, within deallocated LBAs, etc.).
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>No secure erase operation requested</td></tr>
<tr><td>001b</td><td>User Data Erase: All user data shall be erased, contents of the user data after the erase is indeterminate (e.g., the user data may be zero filled, one filled, etc.). The controller may perform a cryptographic erase when a User Data Erase is requested if all user data is encrypted.</td></tr>
<tr><td>010b</td><td>Cryptographic Erase: All user data shall be erased cryptographically. This is accomplished by deleting the encryption key.</td></tr>
<tr><td>011b to 111b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>08</td><td>Protection Information Location (PIL): If set to ‘1’ and protection information is enabled (refer to section 8.3), then protection information is transferred as the first ~~eight~~ bytes of metadata. If cleared to ‘0’ and protection information is enabled, then protection information is transferred as the last ~~eight~~ bytes of metadata. This setting is reported in the 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 versions later than version 1.4 of this 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 enabled and the type of protection information. The values for this field have the following meanings:
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>Protection information is not enabled</td></tr>
<tr><td>001b</td><td>Protection information is enabled, Type 1</td></tr>
<tr><td>010b</td><td>Protection information is enabled, Type 2</td></tr>
<tr><td>011b</td><td>Protection information is enabled, Type 3</td></tr>
<tr><td>100b to 111b</td><td>Reserved</td></tr>
</table>
When end-to-end data protected is enabled, the host shall specify the appropriate protection information in the Read, Verify, Write, or Compare commands.</td></tr>
</table>

Figure 238: Format NVM – Command Dword 10

Bits	Description
04	**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 is not applicable.
03:00	**LBA Format Lower (LBAFL):** This field specifies the least significant 4 bits of the LBA format to apply to the NVM media. This corresponds to the LBA formats indicated in the Identify command, refer to Figure 245 and Figure 246. Only supported LBA formats shall be selected.

Modify a portion of section 6 as shown below:

6 NVM Command Set

…

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 5.23.1.5. If the host does not specify a time limit should be applied, then the controller should apply all error recovery means to complete the operation.

If a host does not set the LBA Format Extension Enable (LBAFEE) field to 1h in the Host Behavior Support feature (refer to section 5.21.1.22), then a controller aborts all I/O commands that access user data to namespaces formatted with (refer to section 8.3.TBD):

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.

Modify portions of section 6.5 as shown below:

6.5 End-to-end Protection Information

The commands that include data transfer may utilize end-to-end data protection. Within these commands, the Pprotection Iinformation Aaction, ~~and~~ Pprotection Iinformation Ccheck, and Storage Tag Check fields are ~~is~~ specified as defined in Figure 355 and Figure FIG_E2EPI.

Figure 355: Protection Information Field Definition

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>03</td><td>Protection Information Action (PRACT): This~~e protection information action~~ bit indicates the action to take for the protection information. This bit is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.
<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~~ Bytes~~</td><td>16B</td><td>The protection information is stripped (read) or inserted (write).</td></tr>
<tr><td>1b</td><td>> 8B~~ Bytes~~</td><td>> 16B</td><td>The protection information is passed (read) or replaces the protection information in ~~first or last 8 bytes of~~ 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. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3
<table>
<tr><th>Bit</th><th>Definition</th></tr>
<tr><td>02</td><td>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 FIG_E2EPI : Storage Tag Check Definition

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>00</td><td>Storage Tag Check: This bit specifies the Storage Tag field, if defined, shall be checked as part of end-to-end processing. If set to ‘1’ enables protection information checking of the Storage Tag field. If cleared to ‘0’, the Storage Tag field is not checked. Refer to section 8.3.

If:
a) the namespace is formatted to use end-to-end protection;
b) the protection information format is (refer to section 8.3.TBD):
○ 16b Guard Protection Information; or
○ 64b Guard Protection Information;
and
c) Storage Tag Size (STS) field is cleared to 0h (refer to Figure FIG_ELBAF),

then this bit is ignored as no Storage Tag field is defined.</td></tr>
</table>

Modify portions of section 6.6 as shown below:

6.6 Compare command

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

If metadata is provided, then a comparison is also performed for the metadata, excluding protection information. The command may specify protection information to be checked as described in section 8.3.1.5.

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 356: Compare – Metadata Pointer

Bits	Description
63:00	**Metadata Pointer (MPTR):** This field contains the Metadata Pointer, if applicable. Refer to Figure 105 for the definition of this field.

Figure 357: Compare – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the data to use for the compare. Refer to Figure 105 for the definition of this field.

Figure FIG_Compare: 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 8.3.TBD.4.1.

Figure 358: 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 359: 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 355. 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 8.3.1.5).
25~~:24~~	Reserved
24	**Storage Tag Check (STC):** This bit specifies the Storage Tag field shall be checked as part of end-to-end processing as defined in Figure FIG_E2EPI.
23:20	Reserved

Figure 359: Compare – Command Dword 12

Bits	Description
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 360: Compare – Command Dword 14

Bits	Description
31:00	~~**Expected Initial Logical Block Reference Tag (EILBRT):** This field specifies the Initial Logical Block Reference Tag expected value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.~~ 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 8.3.TBD.4.1.

Figure 361: Compare – Command Dword 15

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

6.6.1 Command Completion

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

Compare command specific status values are defined in Figure 362.

Figure 362: Compare – Command Specific Status Values

Value	Description
81h	**Invalid Protection Information:** The Protection Information Field (PRINFO) (refer to Figure 359) settings specified in the command are invalid for the Protection Information with which the namespace was formatted (refer to the PI field in Figure 328 and the DPS field in Figure 245) or the EILBRT field is invalid (refer to section 8.3.1.5).

Modify portions of section 6.7.1.1 as shown below:

6.7.1.1 Deallocate

A logical block that has been deallocated using the Dataset Management command is no longer deallocated when the logical block is written. Read operations and Verify operations do not affect the deallocation status of a logical block. The value read from a deallocated logical block shall be deterministic; specifically, the value returned by subsequent reads of that logical block shall be the same until a write operation occurs to that logical block.

The values read from a deallocated logical block and its metadata (excluding protection information) shall be all bytes cleared to 0h (e.g., bits 2:0 in the DLFEAT field are set to 001b), all bytes set to FFh (e.g., bits 2:0 in the DLFEAT field are set to 010b), or the last data written to the associated logical block and its metadata, except that access is prohibited to all data and metadata values written before the most recent successful sanitize operation, if any. The Deallocate Logical Block Features (DLFEAT) field in the Identify

Namespace data structure (refer to Figure 245) may report the values read from a deallocated logical block and its metadata.

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

- have each byte in the Guard field value set to ~~FF~~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 bit in the Application Tag field, Storage Tag field, if defined, ~~value set to FFFFh~~ and the Logical Block Reference Tag, if defined, ~~field~~ value set to ~~FFFFFF~~'1' (indicating the protection information shall not be checked).

Using the Error Recovery feature (refer to section 5.21.1.5), host software may enable an error to be returned if a deallocated or unwritten logical block is read. If this error is supported for the namespace and enabled, then a Read, Verify, or Compare command that includes a deallocated or unwritten logical block shall fail with the Unwritten or Deallocated Logical Block status code. Note: Legacy software may not handle an error for this case.

Note: The operation of the Deallocate function is similar to the ATA DATA SET MANAGEMENT with Trim feature described in ACS-4 and SCSI UNMAP command described in SBC-3.

Modify portions of section 6.TBD from TP 4065b Simple Copy Command as shown below:

6.TBD 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 9999: Copy – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies the data to use for the command. Refer to Figure 105 for the definition of this field.

Figure FIG_Copy: 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 8.3.TBD.4.1, to be used for the write portion of the copy operation.

Figure 9998: 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 9997: 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 355, to be used for the write portion of the copy operation.</td></tr>
<tr><td>25:~~24~~</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 processing as defined in Figure FIG_E2EPI, 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 section 9.1) used for the write portion of the copy operation.</td></tr>
<tr><td>19:16</td><td>Command Extension Type (CETYPE): Specifies the Command Extension Type that applies to the command (refer to section 8.TBD). This field is for the write portion of the copy operation.</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 355, 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 format of the Source Range Entries as follows:
<table>
<tr><th>Code</th><th>Description</th><th>Reference</th></tr>
<tr><td>0h</td><td>The Source Range Entries specify starting LBA, number of logical blocks, and parameters associated with the read portion of the operation when PIF1 bit in the DPC field (refer to Figure 245) is cleared to ‘0’.</td><td>Figure 9993</td></tr>
<tr><td>1h</td><td>The Source Range Entries specify starting LBA, number of logical blocks, and parameters associated with the read portion of the operation when PIF1 bit in the DPC field (refer to Figure 245) is set to ‘1’.</td><td>Figure FIG_F01h</td></tr>
<tr><td>All Others 2h to Fh</td><td colspan="2">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 9996: Copy – Command Dword 13

Bits	Description
31:16	**Directive Specific (DSPEC)**: Specifies the Directive Specific value associated with the Directive Type field (refer to section 9.1).
15:00	Reserved

Figure 9995: Copy – Command Dword 14

Bits	Description
31:00	~~**Initial Logical Block Reference Tag (ILBRT):** This field specifies the Initial Logical Block Reference Tag value for the write portion of the copy operation. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.~~ 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 8.3.TBD.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 9994: 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. ~~This field is only used if the namespace is formatted to use end-to-end protection information.~~ If the namespace is not formatted to use end-to-end protection information, then this field is ignored. Refer to section 8.3.
15:00	**Logical Block Application Tag (LBAT):** This field specifies the Application Tag value for the write portion of the copy operation. ~~This field is only used if the namespace is formatted to use end-to-end protection information.~~ If the namespace is not formatted to use end-to-end protection information, then this field is ignored. Refer to section 8.3.

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

If:

a) the specified Descriptor Format 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 specified Descriptor Format 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 9993 shows the 0h descriptor format and an example with 128 Source Range entries.

Figure 9993: Copy – Source Range Entries Descriptor Format 0h

<table>
<tr><th>Range</th><th>Bytes</th><th>Description</th></tr>
<tr><td rowspan="8">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>31:20</td><td>Reserved</td></tr><tr><td>19:16</td><td><TP 4055>
Command Extension Type (CETYPE): Specifies the Command Extension Type that applies to the command (refer to section 8.TBD). This field is used for the read portion of the copy operation for the LBAs specified in this Source Range entry.
<TP 4055 needs to be red text in NVMe 1_4.Next file></td></tr><tr><td>15:00</td><td>Number of Logical Blocks (NLB): This field indicates the number of logical blocks to be copied. This is a 0’s based value.</td></tr></table></td></tr>
<tr><td>21:20</td><td><from TP 4055>
Command Extension Value (CEV): The definition of this field is dependent on the value of the CETYPE field. Refer to Figure NEW-6 for the definition. This field is used for the read portion of the copy operation for the LBAs specified in this Source Range entry.
<TP 4055 needs to be red text in NVMe 1_4.Next file></td></tr>
<tr><td>23:22</td><td>Reserved</td></tr>
<tr><td>27:24</td><td>~~Expected Initial Logical Block Reference Tag (EILBRT):~~ This field specifies the variable sized Expected Logical Block Storage Tag (ELBST) and Expected Initial Logical Block Reference Tag (ILBRT) fields, which are defined in section 8.3.TBD.4.1, to be ~~This field specifies the Initial Logical Block Reference Tag expected value~~ used for the read portion of the copy operation for the LBAs specified in this Source Range entry. ~~This field is only used if the namespace is formatted to use end-to-end protection information.~~ If the namespace is not formatted to use end-to-end protection information, then this field is ignored. Refer to section 8.3.</td></tr>
<tr><td>29:28</td><td>Expected Logical Block Application Tag (ELBAT): This field specifies the Application Tag expected value used for the read portion of the copy operation for the LBAs specified in this Source Range entry. ~~This field is only used if the namespace is formatted to use end-to-end protection information.~~ If the namespace is not formatted to use end-to-end protection information, then this field is ignored. Refer to section 8.3.</td></tr>
<tr><td>31:30</td><td>Expected Logical Block Application Tag Mask (ELBATM): This field specifies the Application Tag Mask expected value used for the read portion of the copy operation for the LBAs specified in this Source Range entry. ~~This field is only used if the namespace is formatted to use end-to-end protection information.~~ If the namespace is not formatted to use end-to-end protection information, then this field is ignored. Refer to section 8.3.</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>~~EILBRT~~ 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></td><td>4071:4064</td><td>Reserved</td></tr>
</table>

Figure 9993: Copy – Source Range Entries Descriptor Format 0h

Range	Bytes	Description
Source Range 127	4079:4072	Starting LBA
	4083:4080	Read Parameters
	4087:4084	Reserved
	4091:4088	~~EILBRT~~ The variable sized ELBST and EILBRT fields
	4093:4092	ELBAT
	4095:4094	ELBATM

Figure FIG_F01h shows the 01h descriptor format and an example with 102 Source Range entries.

Figure FIG_F01h: Copy – Source Range Entries Descriptor Format 1h

<table>
<tr><th>Range</th><th>Bytes</th><th>Description</th></tr>
<tr><td rowspan="8">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>31:20</td><td>Reserved</td></tr><tr><td>19:16</td><td><TP 4055>
Command Extension Type (CETYPE): Specifies the Command Extension Type that applies to the command (refer to section 8.TBD). This field is used for the read portion of the copy operation for the LBAs specified in this Source Range entry.
<TP 4055 needs to be red text in NVMe 1_4.Next file></td></tr><tr><td>15:00</td><td>Number of Logical Blocks (NLB): This field indicates the number of logical blocks to be copied. This is a 0’s based value.</td></tr></table></td></tr>
<tr><td>21:20</td><td><TP 4055>
Command Extension Value (CEV):The definition of this field is dependent on the value of the CETYPE field. Refer to Figure NEW-6 for the definition. This field is used for the read portion of the copy operation for the LBAs specified in this Source Range entry.
<TP 4055 needs to be red text in NVMe 1_4.Next file></td></tr>
<tr><td>25:20</td><td>Reserved</td></tr>
<tr><td>35:26</td><td>This field specifies the variable sized Expected Logical Block Storage Tag (ELBST) and Expected Initial Logical Block Reference Tag (EILBRT) fields, which are defined in section 8.3.TBD.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.</td></tr>
<tr><td>37:36</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. Refer to section 8.3.</td></tr>
<tr><td>39:38</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. Refer to section 8.3.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td rowspan="8">Source Range 1</td><td>47:40</td><td>Reserved</td></tr>
<tr><td>55:48</td><td>Starting LBA</td></tr>
<tr><td>59:56</td><td>Read Parameters</td></tr>
<tr><td>61:60</td><td>CEV</td></tr>
<tr><td>65:62</td><td>Reserved</td></tr>
<tr><td>75:66</td><td>The variable sized ELBST and EILBRT</td></tr>
<tr><td>77:76</td><td>ELBAT</td></tr>
<tr><td>79:78</td><td>ELBATM</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td rowspan="2">Source Range 101</td><td>4047:4040</td><td>Reserved</td></tr>
<tr><td>4055:4048</td><td>Starting LBA</td></tr>
</table>

Figure FIG_F01h: Copy – Source Range Entries Descriptor Format 1h

Range	Bytes	Description
	4059:4056	Read Parameters
	4061:4060	CEV
	4065:4062	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 247), 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 247), 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 247), then the Copy command shall be aborted with a status code of Command Size Limit Exceeded.

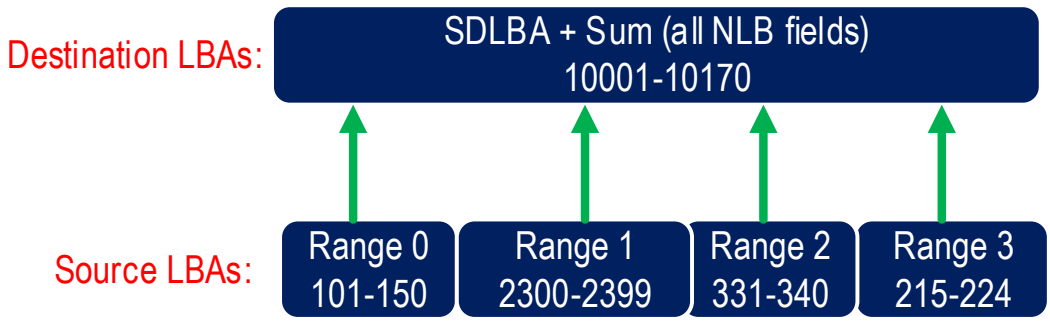
The number of logical blocks written by the Copy command is the sum of all Number of Logical Blocks fields in all Source Range entries specified in the list of Source Range entries.

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 namespace is formatted to use end-to-end protection information, then the protection information is handled as described in section 8.3.

Figure 9992 shows an example of the relationship between the source LBAs and the destination LBAs.

Figure 9992: Source LBA and Destination LBA Relationship Example



6.TBD.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 the Dword 0 of the completion queue entry shall be ~~set~~ cleared to 0h.

Copy command specific errors are defined in Figure 9991.

Figure 9991: 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 359) containing an invalid value for the Protection Information with which the namespace was formatted (refer to the PI field in Figure 328 and the DPS field in Figure 245); • the ILBRT field being invalid (refer to section 8.3.1.5); or • the EILBRT field in a Source Range Entry being invalid (refer to section 8.3.1.5).
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 section 8.19).
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.
87h	**Deallocated or Unwritten Logical Block:** The command failed due to an attempt to copy from an LBA range containing a deallocated or unwritten logical block.

Modify portions of section 6.9 as shown below:

6.9 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 369: Read – Metadata Pointer

Bits	Description
63:00	**Metadata Pointer (MPTR):** This field contains the Metadata Pointer, if applicable. Refer to Figure 105 for the definition of this field.

Figure 370: Read – Data Pointer

Bits	Description
127:00	**Data Pointer (DPTR):** This field specifies where data is transferred to. Refer to Figure 105 for the definition of this field.

Figure FIG_Read: 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 8.3.TBD.4.1.

Figure 371: 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 372: 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 355.
25	Reserved
24	**Storage Tag Check (STC):** This bit specifies the Storage Tag field shall be checked as part of end-to-end processing as defined in Figure FIG_E2EPI.
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 373: Read – Command Dword 13

Bits	Description
31:08	Reserved

Figure 373: 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 374: Read – Command Dword 14

Bits	Description
31:00	~~**Expected Initial Logical Block Reference Tag (EILBRT):** This field specifies the Initial Logical Block Reference Tag expected value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.~~ 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 8.3.TBD.4.1.

Figure 375: Read – Command Dword 15

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

6.9.1 Command Completion

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

Read command specific status values are defined in Figure 376.

Figure 376: 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 372) settings specified in the command are invalid for the Protection Information with which the namespace was formatted (refer to the PI field in Figure 328 and the DPS field in Figure 245) or the EILBRT field is invalid (refer to section 8.3.1.5).

Modify portions of section 6.14 as shown below:

6.14 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 section 5.14.1.2.

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 FIG_Verify: Verify – Command Dword 2 and Dword 3

Bits	Description
63:48	Reserved

Figure FIG_Verify: 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 8.3.TBD.4.1.

Figure 394: Verify – Command Dword 10 and Command Dword 11

Bytes	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 395: 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 355. 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 FIG_E2EPI.
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 396: Verify – Command Dword 14

Bits	Description
31:00	~~**Expected Initial Logical Block Reference Tag (EILBRT):** This field specifies the Initial Logical Block Reference Tag expected value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.~~ 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 8.3.TBD.4.1.

Figure 397: Verify – Command Dword 15

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

6.14.1 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 Figure 125 and to Figure 131.

Verify command specific status values are defined in Figure 398.

Figure 398: Verify – Command Specific Status Values

Value	Description
81h	**Invalid Protection Information:** The Protection Information Field (PRINFO) (refer to Figure 395) settings specified in the command are invalid for the Protection Information with which the namespace was formatted (refer to the PI field in Figure 328 and the DPS field in Figure 245) or the EILBRT field is invalid (refer to section 8.3.1.5).

Modify portions of section 6.15 as shown below:

6.15 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 399: Write – Metadata Pointer

Bits	Description
63:00	**Metadata Pointer (MPTR):** This field contains the Metadata Pointer, if applicable. Refer to Figure 105 for the definition of this field.

Figure 400: 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 Figure 105 for the definition of this field.

Figure FIG_Write: Write – 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 8.3.TBD.4.1.

Figure 401: 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 402: 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 355.
25~~:24~~	Reserved
24	**Storage Tag Check (STC):** This bit specifies the Storage Tag field shall be checked as part of end-to-end processing as defined in Figure FIG_E2EPI.
23:20	**Directive Type (DTYPE)**: Specifies the Directive Type associated with the Directive Specific field (refer to section 9.1).
19:16	Reserved
15:00	**Number of Logical Blocks (NLB):** This field indicates the number of logical blocks to be written. This is a 0's based value.

Figure 403: Write – Command Dword 13

Bits	Description
31:16	**Directive Specific (DSPEC)**: Specifies the Directive Specific value associated with the Directive Type field (refer to section 9.1).
15:08	Reserved

Figure 403: 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 404: Write – Command Dword 14

Bits	Description
31:00	~~**Initial Logical Block Reference Tag (ILBRT):** This field specifies the Initial Logical Block Reference Tag value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.~~ 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 8.3.TBD.4.1.

Figure 405: Write – Command Dword 15

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

6.15.1 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 406.

Figure 406: Write – Command Specific Status Values

Value	Description
80h	**Conflicting Attributes:** The attributes specified in the command are conflicting.
81h	**Invalid Protection Information:** The Protection Information Field (PRINFO) (refer to Figure 402) settings specified in the command are invalid for the Protection Information with which the namespace was formatted (refer to the PI field in Figure 328 and the DPS field in Figure 245) or the ILBRT field is invalid (refer to section 8.3.1.5).
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 section 8.19).

Modify portions of section 6.17 as shown below:

6.17 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 245) and the CDW15.EILBRT, CDW15.ELBATM, and CDW15.ELBAT fields in the Write Zeroes command.

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 6.8.1.1.

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.

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 FIG_WZ: 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 8.3.TBD.4.1.

Figure 410: 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 411: 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 355. 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 processing as defined in Figure FIG_E2EPI. This bit shall be cleared to '0'.
243:16	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 412: Write Zeroes – Command Dword 14

Bits	Description
31:00	~~**Initial Logical Block Reference Tag (ILBRT):** This field indicates the Initial Logical Block Reference Tag value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.~~ 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 8.3.TBD.4.1.

Figure 413: Write Zeroes – Command Dword 15

Bits	Description
31:16	**Logical Block Application Tag Mask (LBATM):** This field indicates the Application Tag Mask value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.
15:00	**Logical Block Application Tag (LBAT):** This field indicates the Application Tag value. This field is only used if the namespace is formatted to use end-to-end protection information. Refer to section 8.3.

6.17.1 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 414.

Figure 414: Write Zeroes – Command Specific Status Values

Value	Description
81h	**Invalid Protection Information:** The Protection Information Field (PRINFO) (refer to Figure 411) settings specified in the command are invalid for the Protection Information with which the namespace was formatted (refer to the PI field in Figure 328 and the DPS field in Figure 245) or the ILBRT field is invalid (refer to section 8.3.1.5).
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 section 8.19).

Modify portions of section 8.3 as shown below:

8.3 End-to-end Data Protection (Optional)

To provide robust data protection from the application to the NVM media and back to the application itself, end-to-end data protection may be used. If this optional mechanism is enabled, then additional protection information (e.g., CRC) is added to the logical block that may be evaluated by the controller and/or host software to determine the integrity of the logical block. This additional protection information, if present, is either the first ~~eight~~ bytes of metadata or the last ~~eight~~ bytes of metadata, based on the format of the namespace. ~~For metadata formats with more than eight bytes, if the protection information is contained within the first eight bytes of metadata~~ If the Metadata Size (refer to Figure 246) 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. ~~For metadata formats with more than eight bytes, if the protection information is contained within the last eight bytes of metadata~~ 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 ~~these last eight bytes~~. As described in section 8.2, metadata and hence this protection information may be configured to be contiguous with the logical block data or stored in a separate buffer.

The most commonly used data protection mechanisms in Enterprise implementations are SCSI Protection Information, commonly known as Data Integrity Field (DIF), and the Data Integrity Extension (DIX). The primary difference between these two mechanisms is the location of the protection information. In DIF, the protection information is contiguous with the logical block data and creates an extended logical block, while in DIX, the protection information is stored in a separate buffer. The end-to-end data protection mechanism defined by this specification is functionally compatible with both DIF and DIX. DIF functionality is achieved by configuring the metadata to be contiguous with logical block data (as shown in Figure 449), while DIX functionality is achieved by configuring the metadata and data to be in separate buffers (as shown in Figure 450).

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 245).

8.3.TBD 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.

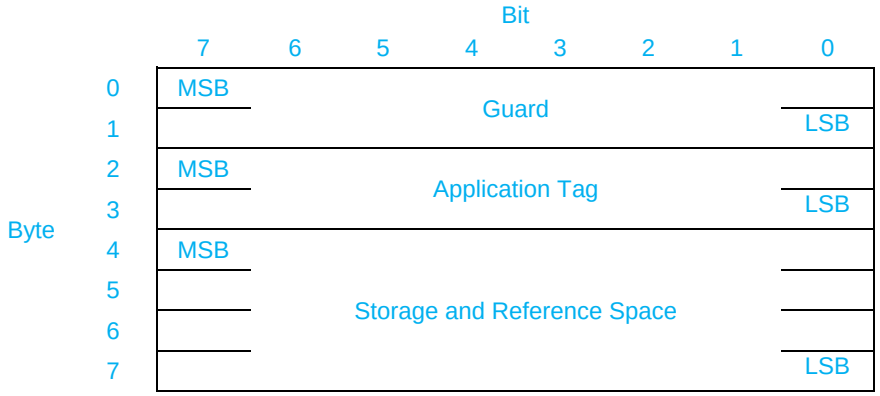
8.3.TBD.1 16b Guard Protection Information

If the Storage Tag Size (STS) field for the LBA Format is cleared to 0h, then t~~T~~the 16b Guard Protection Information ~~format~~ is shown in Figure 451 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 451: 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 FIG_PI16b. The Storage and Reference Space field is separated into a Storage Tag field and a Logical Block Reference Tag field as defined in section 8.3.TBD.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 FIG_PI16b: 16b Guard **Protection Information Format when STS field is non-zero**

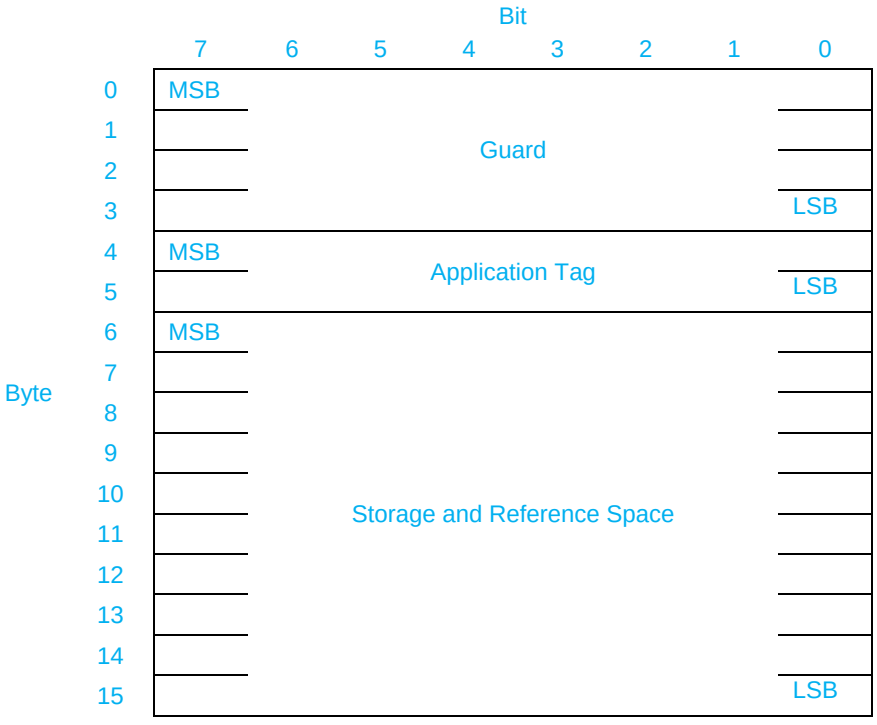


8.3.TBD.2 32b Guard Protection Information

The 32b Guard Protection Information is shown in Figure FIG_PI32b 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 246) 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 8.3.TBD.1).

Figure FIG_PI32b: 32b Guard Protection Information Format



8.3.TBD.2.1 32b CRC Test Cases

Several 32b CRC test cases are shown in Figure FIG_PI32b_Tests.

Figure FIG_PI32b_Tests – 32b CRC Test Cases for 4 KiB Logical Block with no Metadata

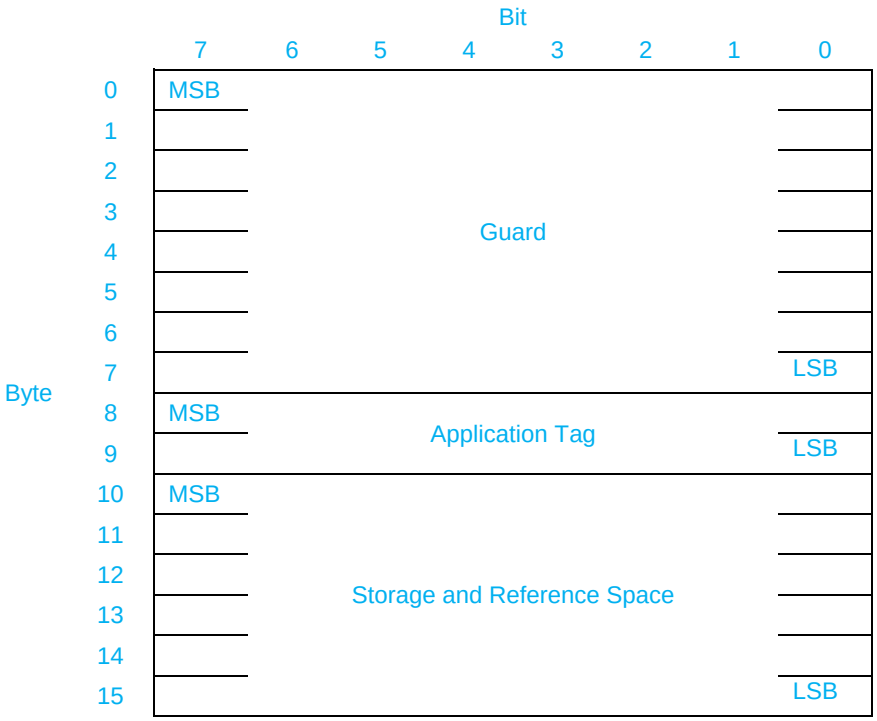
Logical Block Contents	32b Guard Field Value
4 KiB bytes each byte cleared to 00h	98F94189h
4 KiB bytes each byte set to FFh	25C1FE13h
4 KiB bytes of an incrementing byte pattern from 00h to FFh, repeating (e.g. byte 0 is set to 00h, byte 1 is set to 01h, … , byte 254 is set to FEh, byte 255 is set to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

8.3.TBD.3 64b Guard Protection Information

The 64b Guard Protection Information is shown in Figure FIG_PI64b 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 246) 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 FIG_PI64b_Polys. The Application Tag and Storage and Reference Space have the same definition as defined by 16b Guard Protection Information (refer to section 8.3.TBD.1).

Figure FIG_PI64b: 64b Guard Protection Information Format



8.3.TBD.3.1 64b CRC Definition

Figure FIG_PI64b_Polys defines the 64b CRC polynomial used to generate the Guard field for the 64b Guard Protection Information.

Figure FIG_PI64b_Polys – 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.
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.

8.3.TBD.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)):

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

8.3.TBD.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.

8.3.TBD.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 FIG_PI64b 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 CALC_64b_CRC: 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.

8.3.TBD.3.5 64b CRC Test Cases

Several 64b CRC test cases are shown in Figure FIG_PI64b_Tests.

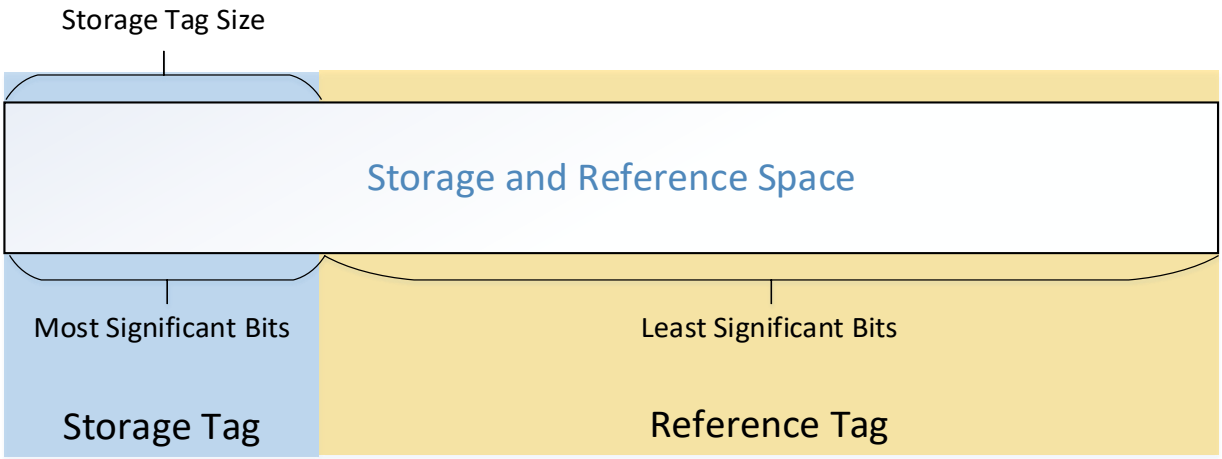
Figure FIG_PI64b_Tests – 64b CRC Test Cases for 4 KiB Logical Block with no Metadata

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

8.3.TBD.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 formats to be separated into a Storage Tag field and a Logical Block Reference Tag field as shown in Figure FIG_ST_RT. 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 FIG_ST_RT: Separation of Storage and Reference Space into Storage Tag and Logical Block Reference Tag



8.3.TBD.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 FIG_STS 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 X2) for each protection information format (refer to section 8.3.TBD).

Figure FIG_STS: 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 FIG_CDWS 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 FIG_CDWS: LBST, ELBST, ILBRT, and EILBRT fields Format in Command Dwords

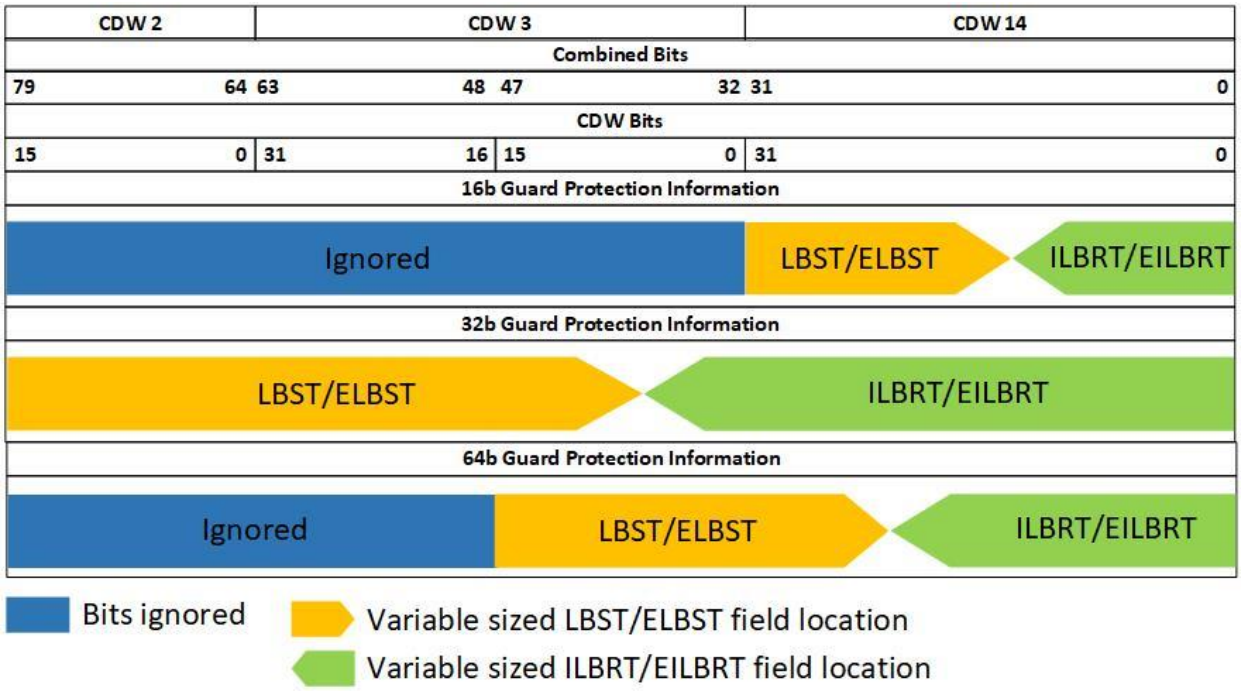


Figure FIG_CMD_CDWS shows the layout of the LBST, ELBST, ILBRT, and EILBRT fields for I/O commands that utilize the fields.

Figure FIG_CMD_CDWS: 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 IST, EIST, ILBRT or EILBRT as defined in Figure FIG_CDWS
32b Guard Protection Information	
Command Dword 2	
15:00	Most significant bit of the LBST or ELBST
Command Dword 3	
31:00	Variable sized IST, EIST, ILBRT or EILBRT as defined in Figure FIG_CDWS
Command Dword 14	
31:00	Variable sized IST, EIST, ILBRT or EILBRT as defined in Figure FIG_CDWS
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 FIG_CDWS
Command Dword 14	
31:00	Variable sized LBST, ELBST, ILBRT or EILBRT as defined in Figure FIG_CDWS

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 246) set to 0h specifying a 512B LBA size;
b) Metadata Size field (refer to Figure 246) set to 8h specifying a 8B metadata size;
c) Protection Information Format field (refer to Figure 245) set to 00b specifying the 16b Guard Protection Information; and
d) Storage Tag Size (STS) field (refer to Figure FIG_ELBAF>) set 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 FIG_16b_WRITE and for a Read command is shown in Figure FIG_16b_READ.

Figure FIG_16b_WRITE: 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 FIG_16b_Read: 16b Guard Protection Information Read Command Example

Bits	Description
Command Dword 2	
15:00	Ignoed
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 246) set to Ch specifying a 4 KiB LBA size;
b) Metadata Size field (refer to Figure 246) set to 10h specifying a 16B metadata size;
c) Protection Information Format field (refer to Figure 245) set to 01b specifying the 32b Guard Protection Information; and
d) Storage Tag Size (STS) field (refer to Figure FIG_ELBAF 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 FIG_32b_WRITE and for a Read command is shown in Figure FIG_32b_READ.

Figure FIG_32b_WRITE: 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 FIG_32b_Read: 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 246) set to Ch specifying a 4 KiB LBA size;
b) Metadata Size field (refer to Figure 246) set to 10h specifying a 16B metadata size;
c) Protection Information Format field (refer to Figure 245) set to 10b specifying the 64b Guard Protection Information; and
d) Storage Tag Size (STS) field (refer to Figure FIG_ELBAF) 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 FIG_64b_WRITE and for a Read command is shown in Figure FIG_64B_READ.

Figure FIG_64b_WRITE: 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 FIG_64b_Read: 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

8.3.1 The PRACT Bit

The protection information processing performed as a side effect of Read and Write commands is controlled by the Protection Information Action (PRACT) bit in the command.

8.3.1.1 Protection Information and Write Commands

Figure 452 provides some examples of the protection information processing that may occur as a side effect of a Write command.

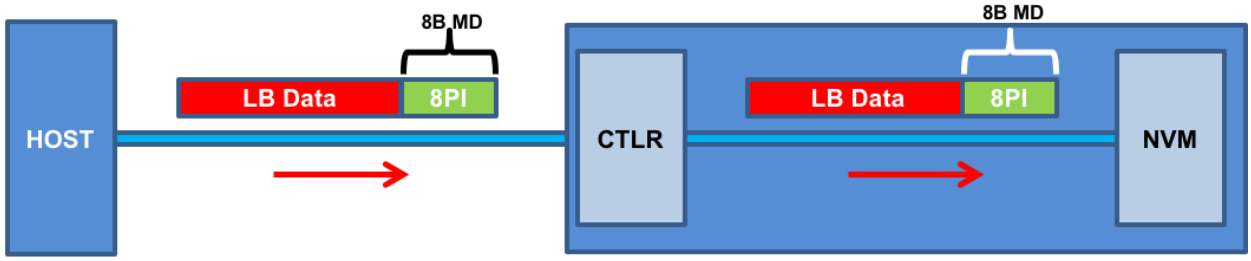
If the namespace is not formatted with end-to-end data protection, then logical block data and metadata is transferred from the host to the NVM with no protection information related processing by the controller.

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).

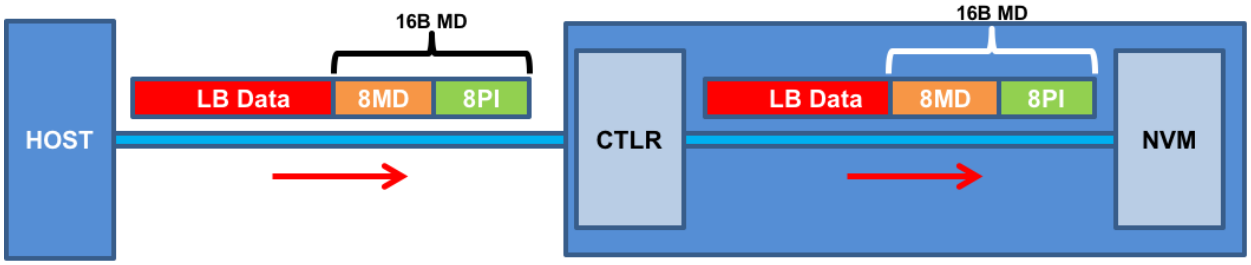
If the namespace is formatted with protection information and the PRACT bit is set to '1', then:

1. If the namespace is formatted with Metadata Size (refer to Figure 246) equal to the protection information size ~~8~~ (refer to ~~Figure 246~~ section 8.3.TBD), 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 ~~8~~, then the logical block data and the metadata are transferred from the host buffer to the controller. As the metadata passes through the controller, the controller overwrites the protection information portion of the metadata. The logical block data and metadata are written to the NVM (i.e., the metadata field remains the same size in the NVM and the host buffer). The location of the protection information within the metadata is configured when the namespace is formatted (refer to the DPS field in Figure 245).

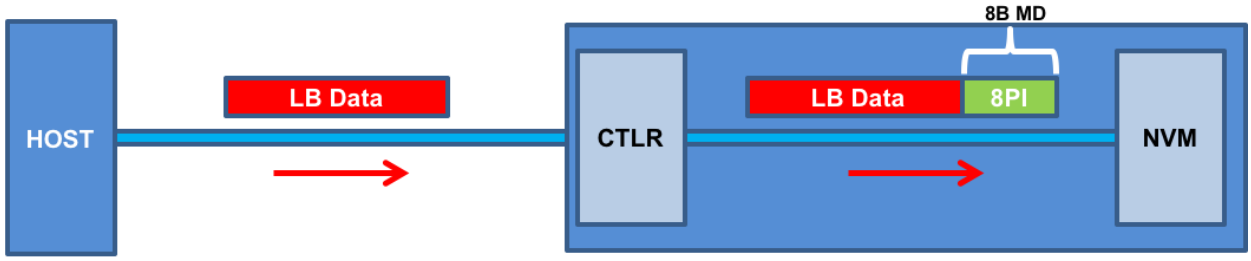
Figure 452: Write Command 16b Guard Protection Information Processing



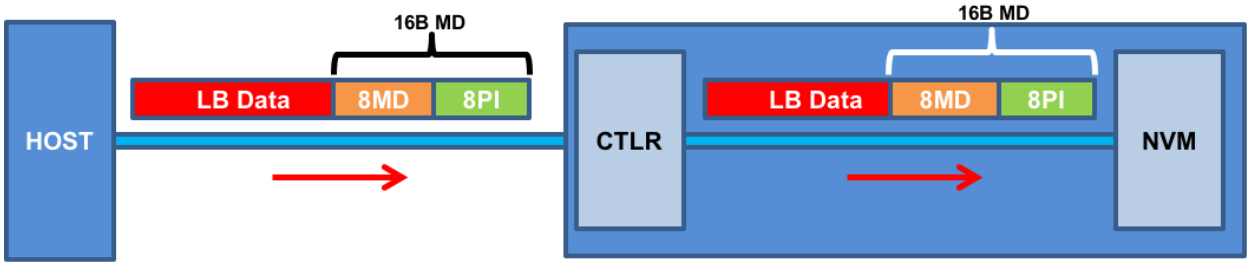
a) MD=8, PI, PRACT=0: Metadata remains same size in NVM and host buffer



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

NOTE: In cases (b) and (d) the Protection Information could be before or after the 8 bytes of metadata.

8.3.1.2 Protection Information and Read Commands

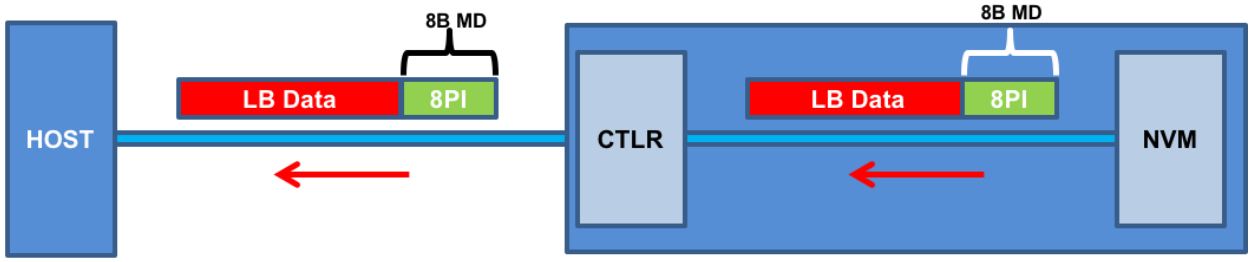
Figure 453 provides some examples of the protection information processing that may occur as a side effect of Read command processing.

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).

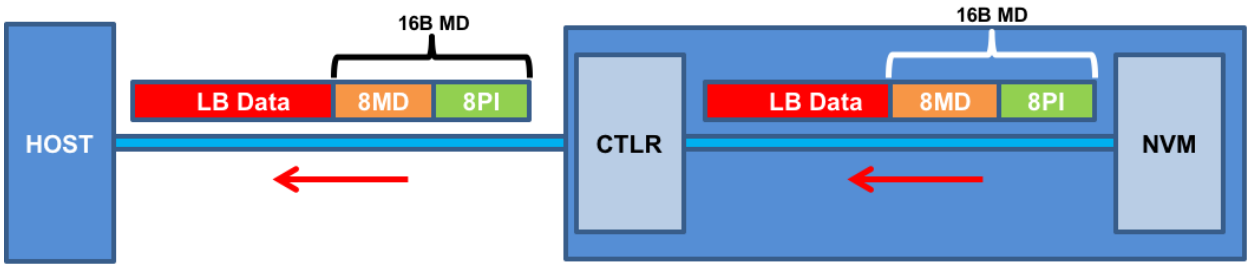
If the namespace is formatted with protection information and the PRACT bit is set to '1', then:

a) if the namespace is formatted with Metadata Size (refer to Figure 246) equal to protection information size ~~8~~ (refer to ~~Figure 246~~ section 8.3.TBD), 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 ~~8~~, the logical block data and the metadata, which in this case contains the protection information and additional host formatted metadata, is read from the NVM by the controller. As the logical block and metadata pass through the controller, the protection information embedded within the metadata is checked. If a protection information check error is detected, the command completes with the status code of the error detected (i.e., End-to-end Guard Check 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).

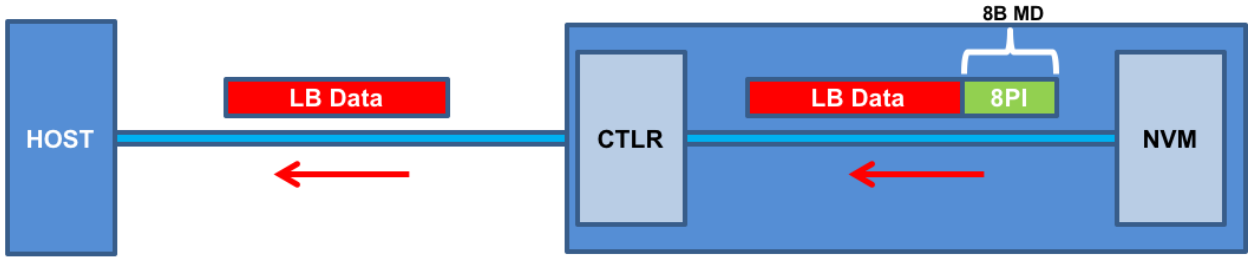
Figure 453: Read Command 16b Guard Protection Information Processing



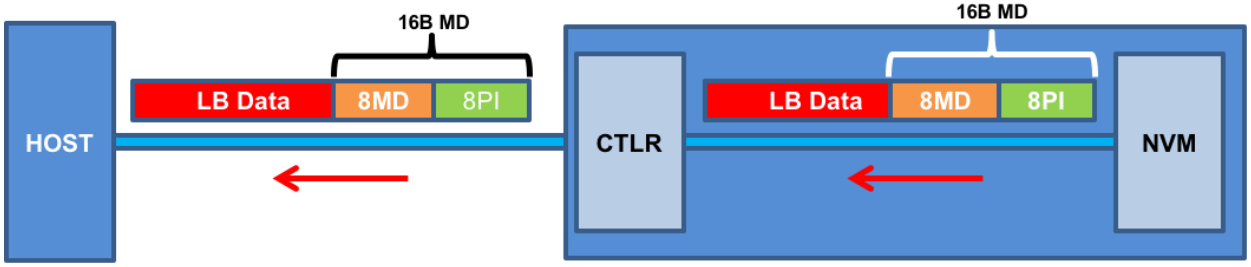
a) MD=8, PI, PRACT=0: Metadata remains same size in NVM and host buffer



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

NOTE: In cases (b) and (d) the PI could be before or after the 8 bytes of metadata.

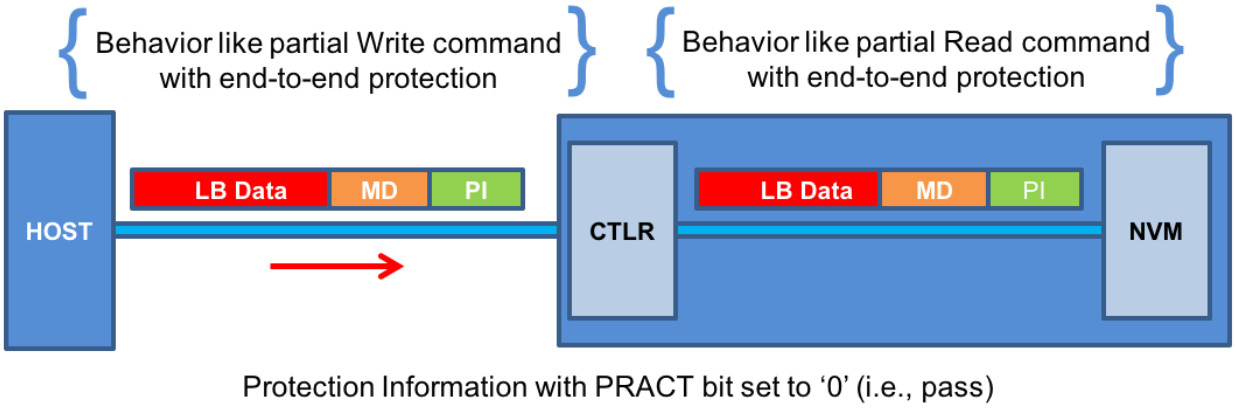
8.3.1.3 Protection Information for Fused Operations

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

8.3.1.4 Protection Information and Compare commands

Figure 454 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 8.3.1.1). 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 8.3.1.2).

Figure 454: Protection Information Processing for Compare



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

8.3.1.5 Control of Protection Information Checking - PRCHK

Checking of protection information consists of the following operations performed by the controller. If the Guard Check bit ~~2~~ of the Protection Information Check (PRCHK) field of the command is set to '1', then the controller compares the protection information Guard field to the CRC for the protection information format (refer to section 8.3.TBD) ~~CRC-16~~ computed over the logical block data. If the Application Tag Check bit ~~1~~ of the PRCHK field is set to '1', then the controller compares unmasked bits in the protection information Application Tag field to the Logical Block Application Tag (LBAT) field in the command. A bit in the protection information Application Tag field is masked if the corresponding bit is cleared to '0' in the Logical Block Application Tag Mask (LBATM) field of the command. If a Storage Tag field is defined in the protection information (refer to section 8.3.TBD.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 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 X2 <from TP 4056>).

If the Reference Tag is defined (refer to Figure FIG_ELBAF), then:

- For Type 1 protection, if the Reference Tag Check bit ~~0~~ of the PRCHK field is set to '1', then the controller compares the ~~protection information~~ Logical Block Reference Tag ~~field~~ to the computed reference tag. The value of the computed reference tag for the first LBA of the command is the value contained in the Initial Logical Block Reference Tag (ILBRT) or Expected Initial Logical Block Reference Tag (EILBRT) field in the command. If the namespace is formatted for Type 1 or Type 2 protection, the computed reference tag is incremented for each subsequent logical block. If the namespace is formatted for Type 3 protection, the reference tag for each subsequent logic block remains the same as the initial reference tag. Unlike SCSI Protection Information Type 1 protection which implicitly uses the least significant four bytes of the LBA, the controller always uses the ILBRT or EILBRT field and requires host software to initialize the ILBRT or EILBRT field to the least

significant bits ~~four bytes~~ of the LBA sized to the number of bits in the Logical Block Reference Tag (refer to section 8.3.TBD.4) when Type 1 protection is used. In Type 1 protection, the controller should check the ILBRT field or the EILBRT field for the correct value; if the value does not match the least significant ~~four bytes~~ bits of the LBA, then the controller completes the command with a status of Invalid Protection Information.

- For Type 2 protection, if the Reference Tag Check bit ~~0~~ of the PRCHK field is set to '1', then the controller compares the ~~protection information~~ Logical Block Reference Tag field from each logical block to the computed reference tag. The computed reference tag is incremented for each subsequent logical block. The value of the computed reference tag for the first LBA of the command is the value contained in the ILBRT or EILBRT field in the command. Host software may set the ILBRT and EILBRT fields to any value.
- For Type 3 protection, if the Reference Tag Check bit ~~0~~ of the PRCHK field is set to '1', then the command should be aborted with status Invalid Protection Information, but may be aborted with status Invalid Field in Command. The controller may ignore the ILBRT and EILBRT fields when Type 3 protection is used because the computed reference tag remains unchanged.
- 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 field, and Logical Block Reference Tag, if defined, ~~fields~~ regardless of the state of the PRCHK field in the command. If the namespace is formatted for Type 1 or Type 2 protection, then all protection information checks are disabled regardless of the state of the PRCHK field when the protection information Application Tag has a value of FFFFh.

If the namespace is formatted for Type 3 protection, then all protection information checks are disabled regardless of the state of the PRCHK field when the protection information Application Tag ~~has a value of FFFFh~~ and ~~the protection information~~ Logical Block Reference Tag, if defined, ~~has~~ have all bits set to '1' ~~a value of FFFFFFFFh~~.

Inserted protection information consists of the computed CRC for the protection information format (refer to section 8.3.TBD) ~~CRC-16~~ 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 ~~field~~.

Modify portions of section 8.3.1.NEW from TP 4065b Simple Copy Command as shown below:

8.3.1.NEW 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 9993), the protection information checks performed by the controller are controlled by the PRINFOR field in Copy command Dword 12 (refer to Figure 9997 and parallels the Read command protection information checks (refer to section 8.3.1.2) 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 9998), the protection information operations performed by the controller are controlled by the PRINFOW field in Copy command Dword 12 (refer to Figure 9997) and parallels the Write command protection information checks (refer to section 8.3.1.1) 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 8.3.1.1.

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.

Modify portions of figure480 in section 8.15 as shown below:

8.15 Sanitize Operations (Optional)

…

Figure 480: Sanitize Operations – Overwrite Mechanism

OIPBP[1]	Overwrite Pass Count[1]	Overwrite Pass Number	Logical Block Data and Non-PI Metadata[2]	Protection Information[3]
'0'	All	All	Overwrite Pattern[1]	Each byte set to ~~FFFFFFFF_FFFFFF~~FFh
'1'	Even	First	Inversion of Overwrite Pattern[1]	Each byte cleared to ~~00000000_000000~~00h
		Subsequent	Inversion of Overwrite Pattern[1] from previous pass (i.e., each bit XORed with '1')	
'1'	Odd	First	Overwrite Pattern[1]	Each byte set to ~~FFFFFFFF_FFFFFF~~FFh
		Subsequent	Inversion of Overwrite Pattern[1] from previous pass (i.e., each bit XORed with '1')	

NOTES:
1. Parameters are specified in Command Dword 10 and Command Dword 11 of the corresponding Sanitize command that started the Overwrite operation. The Overwrite Invert Pattern Between Passes (OIPBP) field is defined in Command Dword 10. The Overwrite Pass Count is defined in Command Dword 10. The Overwrite Pattern is defined in Command Dword 11. Refer to section 5.24.
2. If metadata other than Protection Information is present.
3. If Protection Information is present within the metadata.

Modify portions of NVM Express Zoned Namespace Command Set 1.0 as shown below:

Modify portions of Figure 8 in section 3.1.1 as shown below:

3.1.1 I/O Command Set specific Identify Namespace data structure (CNS TBDah <Refer to the integration of TP 4056>)

Figure 8: I/O Command Set Specific Identify Namespace Data Structure, Zoned Namespaces Command Set

Bytes	O/M [1]	Description
…		
19:16	O	**Finish Recommended Limit (FRL):** Number of seconds before the NVM subsystem may perform the vendor specific action on the zone after the Finish Zone Recommended zone attribute is set to '1'. If this field is cleared to 0h, then no Finish Recommended Limit is reported. Refer to section 5.3.
2815:20		Reserved

Figure 8: I/O Command Set Specific Identify Namespace Data Structure, Zoned Namespaces Command Set

Bytes	O/M [1]	Description
2831:2816	M	**LBA Format 0 Extension (LBAFE0):** This field indicates the LBA format Extension 0 that is supported by the controller. The Zone format field is defined in Figure 9.
2847:2832	O	**LBA Format 1 Extension (LBAFE1):** This field indicates the LBA format 1 Extension that is supported by the controller. The LBA Format Extension field is defined in Figure 9.
~~2863:2848~~	~~O~~	~~**LBA Format 2 Extension (LBAFE2):** This field indicates the LBA format 2 Extension that is supported by the controller. The LBA Format Extension field is defined in Figure 9.~~
~~2879:2864~~	~~O~~	~~**LBA Format 3 Extension (LBAFE3):** This field indicates the LBA format 3 Extension that is supported by the controller. The LBA Format Extension field is defined in Figure 9.~~
~~2895:2880~~	~~O~~	~~**LBA Format 4 Extension (LBAFE4):** This field indicates the LBA format 4 Extension that is supported by the controller. The LBA Format Extension field is defined in Figure 9.~~
~~2911:2896~~	~~O~~	~~**LBA Format 5 Extension (LBAFE5):** This field indicates the LBA format 5 Extension that is supported by the controller. The LBA Format Extension field is defined in Figure 9.~~
~~2927:2912~~	~~O~~	~~**LBA Format 6 Extension (LBAFE6):** This field indicates the LBA format 6 Extension that is supported by the controller. The LBA Format Extension field is defined in Figure 9.~~
~~2943:2928~~	~~O~~	~~**LBA Format 7 Extension (LBAFE7):** This field indicates the LBA format 7 Extension that is supported by the controller. The LBA Format Extension field is defined in Figure 9.~~
~~2959:2944~~	~~O~~	~~**LBA Format 8 Extension (LBAFE8):** This field indicates the LBA format 8 Extension that is supported by the controller. The LBA Format Extension field is defined in Figure 9.~~
~~2975:2960~~	~~O~~	~~**LBA Format 9 Extension (LBAFE9):** This field indicates the LBA format 9 Extension that is supported by the controller. The LBA Format Extension field is defined in Figure 9.~~
~~2991:2976~~	~~O~~	~~**LBA Format 10 Extension (LBAFE10):** This field indicates the LBA format Extension 10 that is supported by the controller. The LBA Format Extension field is defined in Figure 9.~~
~~3007:2992~~	~~O~~	~~**LBA Format 11 Extension (LBAFE11):** This field indicates the LBA format Extension 11 that is supported by the controller. The LBA Format Extension field is defined in Figure 9.~~
~~3023:3008~~	~~O~~	~~**LBA Format 12 Extension (LBAFE12):** This field indicates the LBA format Extension 12 that is supported by the controller. The LBA Format Extension field is defined in Figure 9.~~
~~3039:3024~~	~~O~~	~~**LBA Format 13 Extension (LBAFE13):** This field indicates the LBA format Extension 13 that is supported by the controller. The LBA Format Extension field is defined in Figure 9.~~
~~3055:3040~~	~~O~~	~~**LBA Format 14 Extension (LBAFE14):** This field indicates the LBA format Extension 14 that is supported by the controller. The LBA Format Extension field is defined in Figure 9.~~
~~3071:3056~~	~~O~~	~~**LBA Format 15 Extension (LBAFE15):** This field indicates the LBA format Extension 15 that is supported by the controller. The LBA Format Extension field is defined in Figure 9.~~
…		
3839:3824	O	**LBA Format 63 Extension (LBAFE63):** This field indicates the LBA format Extension 63 that is supported by the controller. The LBA Format Extension field is defined in Figure 9.
4095:3840	O	Vendor Specific
NOTES: 1. O/M definition: O = Optional, M = Mandatory.		

Modify portions of section 4.5 as shown below:

4.5 Zone Append command

The Zone Append command writes data and metadata, if applicable, to the I/O controller for the zone indicated by the ZSLBA field. The controller assigns the data and metadata, if applicable, to a set of logical blocks within the zone. The lowest LBA of the set of logical blocks written is returned in the completion queue entry (refer to section 4.5.1). The host may also specify protection information to include as part of the operation.

This 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.

Write ordering in the case of multiple outstanding Zone Append commands to a zone is undefined and left to the controller.

If the zone which the Zone Append command specifies is not of zone type Sequential Write Required, then the command shall be aborted with a status code of Invalid Field in Command.

If the ZSLBA field in the Zone Append command does not specify the lowest logical block for a zone, then the command shall be aborted with a status code of Invalid Field in Command.

The AWUN, NAWUN, NABSN, AWUPF, NAWUPF, NABSPF atomicity parameters apply as defined in the Atomic Operations section in the NVMe Base specification to the Zone Append command.

Figure 40: Zone Append – Metadata Pointer

Bits	Description
63:00	**Metadata Pointer (MPTR):** This field contains the Metadata Pointer, if applicable. Refer to the NVMe Base specification for the definition of this field.

Figure 41: Zone Append – 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 NVMe Base specification for the definition of this field.

Figure FIG_ Append: Zone Append – 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 are defined in the NVMe Base specification.

Figure 42: Zone Append – Command Dword 10 and Command Dword 11

Bits	Description
63:00	**Zone Start Logical Block Address (ZSLBA):** This field indicates the 64-bit address of the lowest logical block of the zone in which the data and metadata, if applicable, associated with this command is to be stored. Command Dword 10 contains bits 31:00; Command Dword 11 contains bits 63:32.

Figure 43: Zone Append – 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, as defined in the NVMe Base specification.

Bits	Description
30	**Force Unit Access (FUA):** If set to '1', then for data and metadata, if any, associated with logical blocks specified by the Zone Append 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, as defined in the NVMe Base specification.
29:26	**Protection Information Field (PRINFO):** Specifies the Protection Information field, as defined in the NVMe Base specification.
25	**Protection Information Remap PIREMAP):** This bit determines the contents of the reference tag written to the media (refer 4.5). For Type 1 protection, the controller shall abort the command with a status code of Invalid Protection Information if this bit is cleared to '0'. For Type 3 protection, the controller shall abort the command with a status code of Invalid Protection Information if this bit is set to '1'.
24	~~Reserved~~**Storage Tag Check (STC):** This bit specifies the Storage Tag field shall be checked as part of end-to-end processing as defined in the NVMe Base specification.
23:20	**Directive Type (DTYPE)**: Specifies the Directive Type associated with the Directive Specific field (refer to 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 FIG_Append_13: Zone Append – Command Dword 13

Bits	Description
31:16	**Directive Specific (DSPEC)**: Specifies the Directive Specific value associated with the Directive Type field (refer to the NVMe Base specification).
15:00	Reserved

Figure 44: Zone Append – Command Dword 14

Bits	Description
31:00	~~**Initial Logical Block Reference Tag (ILBRT):** This field specifies the Initial Logical Block Reference Tag value. This field is ignored if the zoned namespace is not formatted to use end-to-end protection information. Refer to section 4.5.2.~~ The variable sized Logical Block Storage Tag (LBST) and Initial Logical Block Reference Tag (ILBRT) fields are defined in the NVMe Base specification.

Figure 45: Zone Append – Command Dword 15

Bits	Description
31:16	**Logical Block Application Tag Mask (LBATM):** This field specifies the Application Tag Mask value. This field is ignored if the zoned namespace is not formatted to use end-to-end protection information. Refer to the End-to-end Data Protection section in the NVMe Base specification.
15:00	**Logical Block Application Tag (LBAT):** This field specifies the Application Tag value. This field is ignored if the zoned namespace is not formatted to use end-to-end protection information. Refer to the End-to-end Data Protection section in the NVMe Base specification.