LEGAL NOTICE:

LEGAL DISCLAIMER:

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

NVM Express™ Technical Errata

Errata ID	**004**
Revision Date	**2021-02-25**
Affected Spec Ver.	**NVM Express 1.4b**
Corrected Spec Ver.	**NVM Express 1.4+**

Errata Author(s)

Company	Authors
Broadcom	Brad Besmer
Dell EMC	Austin Bolen, David Black
HP Enterprise	Curtis Ballard
Intel	Mike Allison, Kalyan Sanagavarapu
Micron	Walt Hubis
NetApp	Fred Knight
Samsung	Judy Brock
Seagate	Jim Hatfield
Western Digital	Yoni Shternhell, Christoph Hellwig

Errata Overview

Miscellaneous corrections and clarifications to NVMe 1.4b and NVM 1.4_NEXT

Revision History

Revision Date	Change Description
2020-03-20	1) Took all ECN 004 suggestions to date and put them in order 2) Added context from NVMe 1.4 as needed 3) Indicated which items need further discussion to create a specific change request 4) Indicated some items that may belong in ECN 005 instead
2020-03-25	1) Added change for section 8.4 from Yoni Shternhell 2) Added change to section 5.21.1 from Fred Knight, et al.
2020-04-09	1) Added change for 8.21.3 for Matt Goepfert and Fred Knight 2) Added change for 8.3.1.5 for Paul Suhler

Revision Date	Change Description
2020-04-24	1) Added changes for figure 104, Figure 245, 8.25.1, Figure 68, 5.21.1.14, and Figure 84 2) Removed items that moved to ECN005 3) Removed items that were already in ECN001 and incorporated in NVMe 1.4a 4) Recorded decisions of ad hoc meeting 4/24/2020 5) Re-baselined text, references, etc. to NVMe 1.4a 6) Removed items that are already in NVMe 1.4a or are covered in other TPs: Figure 84, 5.13, Figure 194, 5.14.1.10, 5.14.1.11, 5.14.1.12, 5.22, 6.1.6, 6.7.1.1 7) Removed items to be covered in future ECNs: a) Consistency (status of xx, status code of xx, status set to,…) b) Consistency ('namespace id', 'namespace identifier', 'NSID')
2020-05-01	1) Resolved open issues: 8.12.1, 8.25.1 2) Added new changes for: 1.6.TBD, 3.1.18, 3.1.19, 3.1.20, figure 205, figure 245, figure 247 3) Agreed to defer these changes to a future ECN or TP: 1.6.tbd, 3.1.18, 3.1.19, 3.1.20, 5.8, 6.4.1
2020-05-02	1) Removed changes that we agreed to defer 2) Changed the summary of changes from a table to a list
2020-06-22	1) Misc cleanup to prepare for member review
2020-06-24	2) Updated Walt Hubis' changes to figure 245, figure 247 and section 8.25.1
2020-07-07	1) Added changes to 7.8 and table 184 2) Added change to figure 249 (No-Deallocate Inhibited)
2020-08-14	1) Added changes to figure 91, figure 148, section 4.5, figure 247, figure 249 2) Counted the number of instances of No-Deallocate After Sanitize with/without the hyphen 3) Added figure 114
2020-09-10	1) In the 9/10/2020 WG meeting, we agreed on the resolution of changes: a. figure 114 and figure 127 as not backward compatible, b. section 4.5, c. to hyphenate all instances of 'No Deallocate After Sanitize" (figure 316, figure 332, and section 8.15) d. to add a table footnote for all references to (VSIL, EL, and VSEDL) in the Persistent Event log (figure 228 and figure 241) i. Note: this modifies the existing NVMe Base 1.4_NEXT 2020.07.21.docx e. modify figure 105 and figure and mark it as not backward compatible 2) We also agreed to stop accepting new material for this ECN and start the approval process.
2020-11-03	Organized changes and Integrated into the NVMe Base Specification
2020-12-15	1) Synchronized section/figure numbers with NVMe 1.4b, and renamed 2) Reordered some sections to be in section number order 3) Moved editors notes: from (before section number) to(after section number) 4) Figure 117: changed 'memory' to 'host memory' in one missed location
2020-12-17	1) Added missing section numbers. Aligned to public version of NVMe 1.4b for figure numbers. 2) Review accepted blue text change to section 8.25.1
2021-01-02	1) Made lists in the NDI field definition.
2021-01-19	1) John Geldman reworded the NDI text to make it clear that the NDI bit has no effect on a Sanitize command if the No-Deallocate After Sanitize bit is cleared to '0'.
2021-01-27	1) Integrated into the NVMe Base Specification

Revision Date	Change Description
2021-02-03	1) Reverted changes in the Select field, Estimated Time For Overwrite, Estimate Time For Block Erase, Estimate Time For Crypto Erase, Estimated Time For Overwrite With No-Deallocate Media Modification, Estimated Time For Block Erase With No-Deallocate Media Modification, and **Estimated Time For Crypto Erase With No-Deallocate Media Modification due to late comments that need more time to review.**
2021-02-11	1) Accepted all changes and removed all comments. Removed page breaks not needed.
2021-02-16	1) Integrated into the NVMe Base Specification
2021-02-18	1) Updated the Errata Overview to spell out the words and reference NVMe 1.4b.
2021-02-25	2) Removed comments, accepted all changes, converted all references/cross-references to text.

Incompatible Changes

- In figure 115 (SGL Descriptor Sub Type Values), the controller is now required to report an error if the SGL Descriptor Sub Type is 1h and the SGL Descriptor Type is 1h or 4h. The handling of these cases was previously unspecified.
- In figure 128, thee description of the status code SGL Descriptor Type Invalid has been extended to include the combination of type and subtype as well.
- In figure 106 (Command Format – Admin and NVM Command Set) handles a previously unspecified case: namespace identifier is not user for the command and the host specifies a NSID set to FFFFFFFFh.

Summary of changes:

ECN 004: This errata includes:

- Miscellaneous editorial font, capitalization, spelling, references, punctuation, figure title changes, and use of standard phrases
- Adding references
- Clarifying whether the PMRCAP, PMREBS, PMRSWTP, CMBMSC, and CMBSTS controller registers are optional or not
- Removing mention of any explicit NVMe-oF specification revision
- Add specification of a previously unspecified use of NSID=FFFFFFFFh
- Clarifying the combinations of SGL Descriptor Type and Subtype that are valid
- Clarifying that the Firmware Revision is an ASCII string
- Clarifying what is returned by Get Features after a controller level reset
- Clarifying the intent of User Data Erase in the Format NVM command
- Clarifying the sequencing of fused operations
- Clarifying the summary description of the metadata region
- Clarify the use of the NOWS field
- Clarify non-security related uses of the Timestamp feature
- Renumber section 8.3.1.5 as 8.3.2: Control of Protection Information Checking (PRCHK)
- Clarify that power measurement methods are out of scope
- Remove a SHALL use in an informative section
- Clarifying some parts of the Sanitize feature regarding the No-Deallocate After Sanitize bit
- Add table footnotes to provide references for the EL and VSIL fields in the Persistent Event log

Description of Specification Changes

Markup Conventions:

Black:	Unchanged (however, hot links are removed)
~~Red Strikethrough~~:	Deleted
Blue underscore:	New
Red Highlighted:	TBD values, anchors, and links to be inserted.

<Green Bracketed>: Notes to editor

Specific changes

1.6.8 command submission

<remove the explicit revision number>

For NVMe over PCIe implementations, a command is submitted when a Submission Queue Tail Doorbell write has completed that moves the Submission Queue Tail Pointer value past the Submission Queue slot in which the command was placed.

For NVMe over Fabrics implementations, refer to section 1.4.14 in the NVMe over Fabrics ~~revision 1.0~~ specification.

3.1 Register Definition

<mark some registers optional>

Figure 68: Register Definition

Start	End	Symbol	Description
…			
E00h	E03h	PMRCAP	Persistent Memory Capabilities ~~(Optional)~~
E04h	E07h	PMRCTL	Persistent Memory Region Control (Optional)
E08h	E0Bh	PMRSTS	Persistent Memory Region Status (Optional)
E0Ch	E0Fh	PMREBS	Persistent Memory Region Elasticity Buffer Size (Optional)
E10h	E13h	PMRSWTP	Persistent Memory Region Sustained Write Throughput (Optional)

…

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

<mark register optional>

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

…

3.1.17 Offset 58h: CMBSTS – Controller Memory Buffer Status

<mark register optional>

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

…

3.1.18 Offset E00h: PMRCAP – Persistent Memory Region Capabilities

<modify figure 91 PMRTU as noted>

Figure 91: Offset E00h: PMRCAP – Persistent Memory Region Capabilities

<table>
<tr><th>Bits</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>…</td><td></td><td></td><td>…</td></tr>
<tr><td>9:8</td><td>RO</td><td>Impl Spec</td><td>Persistent Memory Region Time Units (PMRTU): Indicates Persistent Memory Region time units.
<table><tr><th>Value</th><th>Persistent Memory Region Time Units</th></tr><tr><td>00b</td><td>500 milliseconds</td></tr><tr><td>01b</td><td>minutes</td></tr><tr><td>~~01b~~ 10b to 11b</td><td>Reserved</td></tr></table></td></tr>
<tr><td></td><td></td><td></td><td>…</td></tr>
</table>

<and also: apply changes from TP 4000a. The red text is what was in the original TP.>

Figure 91: Offset E00h: PMRCAP – Persistent Memory Region Capabilities

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

<mark register optional>

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

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

…

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

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

…

4.1 Submission Queue & Completion Queue Definition

<remove the explicit revision number>

Sections 4.1, 4.1.1, and 4.1.2 apply to NVMe over PCIe implementations only. For NVMe over Fabrics implementations, refer to section 2.4 and the subsections of that section in the NVMe over Fabrics ~~revision 1.0~~ specification.

…

4.2 Submission Queue Entry – Command Format

<minor rewording>

Figure 105: Command Dword 0

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td></td><td>…</td></tr>
<tr><td>15:14</td><td>PRP or SGL for Data Transfer (PSDT): This field specifies whether PRPs or SGLs are used for any data transfer associated with the command. PRPs shall be used for all Admin commands for NVMe over PCIe implementations. SGLs shall be used for all Admin and I/O commands for NVMe over Fabrics implementations (i.e., this field set to 01b). ~~This field shall be set to 01b for NVMe over Fabrics revision 1.0 implementations.~~ The definition is described in the table below.
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>PRPs are used for this transfer.</td></tr>
<tr><td>01b</td><td>SGLs are used for this transfer. If used, Metadata Pointer (MPTR) contains an address of a single contiguous physical buffer that is byte aligned.</td></tr>
<tr><td>10b</td><td>SGLs are used for this transfer. If used, Metadata Pointer (MPTR) contains an address of an SGL segment containing exactly one SGL Descriptor that is qword aligned.</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table>
If there is metadata that is not interleaved with the logical block data, as specified in the Format NVM command, then the Metadata Pointer (MPTR) field is used to point to the metadata. The definition of the Metadata Pointer field is dependent on the setting in this field. Refer to Figure 105.</td></tr>
<tr><td></td><td>…</td></tr>
</table>

<modify part of figure 106 to handle a previously unspecified case>

<this change is not backward compatible>

Figure 106: 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 140, Figure 141, and Figure 348 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 140 and Figure 141), 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~~ FFFFFFFFh, then the controller should abort the command with status Invalid Field in Command, unless otherwise specified.

4.4 Scatter Gather List (SGL)

<modify figure 115>
<this change is not backward compatible>

Figure 115: SGL Descriptor Sub Type Values

~~SGL Descriptor Types~~	~~SGL Descriptor Sub Type~~	~~Sub Type Description~~
~~0h, 2h, 3h, 4h~~	~~0h~~	~~**Address:** The Address field specifies the starting 64-bit memory byte address of the Data Block, Segment, or Last Segment descriptor~~
~~0h, 2h, 3h~~	~~1h~~	~~**Offset:** The Address field contains an offset from the beginning of the location where data may be transferred. For NVMe over PCIe implementations, this Sub Type is reserved. For NVMe over Fabrics implementations, refer to the NVMe over Fabrics specification for details on the location from which the offset is specified.~~
~~All~~	~~Ah to Fh~~	~~**NVMe Transport Specific:** The definitions for this range of Sub Types are defined by the binding section for the associated NVMe Transport.~~

SGL Descriptor Sub Type	SGL Descriptor Types	Sub Type Description
0h	0h, 2h, 3h, 4h	**Address:** The Address field specifies the starting 64-bit memory byte address of the Data Block, Segment, or Last Segment descriptor.
	1h	For Type 1h, the Sub Type field shall be cleared to 0h.
	All others	Reserved
1h	0h, 2h, 3h	**Offset**: The Address field contains an offset from the beginning of the location where data may be transferred. For NVMe over PCIe implementations, this Sub Type is reserved. For NVMe over Fabrics implementations, refer to the NVMe over Fabrics specification for details on the location from which the offset is specified.
	1h	The controller shall abort the command with the status of SGL Descriptor Type Invalid.
	4h	The controller shall abort the command with the status of SGL Descriptor Type Invalid.
	All others	Reserved
Ah to Fh	All	**NVMe Transport Specific:** The definitions for this range of Sub Types are defined by the binding section for the associated NVMe Transport.
All others	All	Reserved

<add clarifying text to figure 117>

Figure 117: SGL Bit Bucket descriptor

Bytes	Description
07:00	Reserved
11:08	**Length:** The Length field specifies the amount of source data that is discarded. An SGL Bit Bucket descriptor specifying that no source data be discarded (i.e., the length field cleared to 0h) is a valid SGL Bit Bucket descriptor. If the SGL Bit Bucket descriptor describes a destination data buffer (e.g., a read from the controller to host memory), then the Length field specifies the number of bytes of the source data which the controller shall discard (i.e., not transfer to the destination data buffer). If the SGL Bit Bucket descriptor describes a source data buffer (e.g., a write from host memory to the controller), then the Bit Bucket Descriptor shall be treated as if the Length field were cleared to 0h (i.e., the Bit Bucket Descriptor has no effect). If SGL Bit Bucket descriptors are supported, their length in a destination data buffer shall be included in the specified length of data to be transferred (e.g., their length in a source data buffer is not included in the NLB parameter).

4.5 Metadata Region (MR)

<Modify section 4.5>

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

4.6.1.2.1 Generic Command Status Definition

…

<modify the definition of SGL Descriptor Type Invalid in figure 127>

<this change is not backward compatible>

Figure 128: Status Code – Generic Command Status Values

Value	Description
11h	**SGL Descriptor Type Invalid:** The type of an SGL Descriptor is a type that is not supported by the controller, or the combination of type and subtype is not supported by the controller.
…	**…** <change font size of reference to Figure 128 from Arial 10 to Arial 9>
20h	**Namespace is Write Protected:** The command is prohibited while the namespace is write protected as a result of a change in the namespace write protection state as defined by the Namespace Write Protection State Machine (refer to Figure 489).

4.12 Fused Operations

<clarify the sequencing of fused operations>

Fused operations enable a more complex command by “fusing” together two simpler commands. This feature is optional; support for this feature is indicated in FUSES field in the Identify Controller data structure in Figure 251. In a fused operation, the requirements are:

- The commands shall be executed in sequence as an atomic unit. The controller shall behave as if no other operations have been executed between these two commands;
- The operation ends at the point an error is encountered in either command. If the first command in the sequence failed, then the second command in the sequence shall be aborted. If the second command in the sequence failed, then the completion status of the first command is sequence specific;

…

4.13 Command Arbitration

<remove the explicit revision number>

For NVMe over PCIe implementations, a command is submitted to the controller when a Submission Queue Tail Doorbell write by the host moves the Submission Queue Tail Pointer past the slot containing the corresponding Submission Queue entry. For NVMe over Fabrics implementations, refer to section 1.4.14 in the NVMe over Fabrics ~~revision 1.0~~ specification for the definition of command submission. The controller transfers submitted commands into the controller for subsequent processing using a vendor specific algorithm.

5 Admin Command Set

<change capitalization of 'section' >

The Admin Command Set defines the commands that may be submitted to the Admin Submission Queue.

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

Admin commands should not be impacted by the state of I/O queues (e.g., a full I/O Completion Queue should not delay or stall the Delete I/O Submission Queue command).

Figure 141 defines Admin commands while Figure 142 defines I/O Command Set Specific Admin commands that are specific to the NVM Command Set (i.e., NVM Command Set Specific Admin commands). Refer to section ~~Section~~ 7.1 for mandatory, optional, and prohibited commands for the various controller types.

5.2 Asynchronous Event Request command

…

<change 'see' to 'refer to'>

The following event types are defined:

a) **Error event:** Indicates a general error that is not associated with a specific command (refer to Figure 147). To clear this event, host software reads the Error Information log (refer to section 5.14.1.1) using the Get Log Page command with the Retain Asynchronous Event bit cleared to '0';
b) **SMART / Health Status event:** Indicates a SMART or health status event (refer to Figure 148). To clear this event, host software reads the SMART / Health Information log (refer to section 5.14.1.2) using the Get Log Page command with the Retain Asynchronous Event bit cleared to '0'. The SMART / Health conditions that trigger asynchronous events may be configured in the Asynchronous Event Configuration feature using the Set Features command (refer to section 5.21);
c) **Notice event:** Indicates a general event (refer to Figure 149). To clear this event, host software reads the appropriate log page as described in Figure 149. The conditions that trigger asynchronous events may be configured in the Asynchronous Event Configuration feature using the Set Features command (refer to ~~see~~ section 5.21.1.11). These notice events include:

…

5.2.1 Command Completion

<modify figure 149>

06h	**Endurance Group Event Aggregate Log Page Change:** Indicates that event entries for one or more Endurance Groups (refer to section 5.14.1.9) have been added to the Endurance Group ~~Predictable Latency~~ Event Aggregate log. To clear this event, the host issues a Get Log Page command with the Retain Asynchronous Event bit cleared to '0' for the Endurance Group Event Aggregate log.

5.13 Get Features command

…

<Modify Figure 185 >

Figure 185: Get Features – Command Dword 10

Bits	Description
31:11	Reserved
10:08	**Select (SEL):** This field specifies which value of the attributes to return in the provided data: Select / Description: 000b – Current 001b – Default 010b – Saved 011b – Supported Capabilities 100b to 111b – Reserved Refer to section 5.13.1 and section 7.8 for details on the value returned in each case. The controller indicates in bit 4 of the Optional NVM Command Support field of the Identify Controller data structure in Figure 251 whether this field is supported. If a Get Features command is received with the Select field set to 010b (i.e., saved) and the controller does not support the Feature Identifier being saved or does not currently have any saved values, then the controller shall operate as if the Select field is set to 001b (i.e., default).
07:00	**Feature Identifier (FID):** This field specifies the identifier of the Feature for which to provide data.

5.14.1 Log Specific Information

<correct capitalization of the word 'log pages>

Figure 195 and Figure 196 define the log pages that may be retrieved with the Get Log Page command and the scope of the information that is returned in those log pages. Refer to section 7.1 for mandatory, optional, and prohibited log pages for the various controller types.

Log pages that indicate a scope of NVM subsystem return information that is global to the NVM subsystem. Log pages that indicate a scope of controller return information that is specific to the controller that is processing the command. Log pages that indicate a scope of Namespace return information that is specific to the specified namespace. For log pages that indicate multiple scopes, the namespace identifier that is specified determines which information is returned. The definition of any individual field within a log page may indicate a different scope that is specific to that individual field.

For log pages with a scope of NVM subsystem or controller (as shown in Figure 195 and Figure 196), the controller should abort commands that specify namespace identifiers other than 0h or FFFFFFFFh with status Invalid Field in Command. Otherwise the rules for namespace identifier usage in Figure 106 apply.

…

5.14.1.8 Telemetry Controller-Initiated (Log Identifier 08h)

…

<use consistent words for power cycle>

Figure 207: Get Log Page – Telemetry Controller-Initiated Log (Log Identifier 08h)

	…
383	**Telemetry Controller-Initiated Data Generation Number:** Contains a value that is incremented each time the controller initiates a capture of its internal controller state into the Telemetry Controller-Initiated Data Blocks. If the value of this field is FFh, then the field shall be cleared to 0h when incremented (i.e., rolls over to 0h). This field is persistent across power cycles ~~on~~.
	…

5.14.1.13.1.4 Power-on or Reset Event (Event Type 04h)

<Add a table footnote to figure 221 (as modified in figure 228 NVMe Base 1.4_NEXT 2020.07.21.docx>
<and make the first column wider to remove wrapping >

Figure 221: Power-on or Reset Event (Event Type 04h)

Bytes	Description
07:00	**Firmware Revision:** Contains the firmware revision that becomes effective when CC.EN transitions from '0' to '1'.
EL-VSIL-1:08 NOTE 1	**Reset Information List:** Contains a list of one or more Controller Reset Information descriptors (refer to Figure 222). If virtualization management is not implemented, then the list shall contain a Controller Reset Information descriptor for every controller in the NVM subsystem. If virtualization management is implemented, then the list shall contain a Controller Reset Information descriptor for every primary controller. The Controller Reset Information descriptor is shown in Figure 222. ~~EL is the value from the Event Length field in the Persistent Event Log event header (refer Figure 216) and VSIL is the value from the Vendor Specific Information Length field in the Persistent Event Log header.~~
NOTE: **1.** Refer to Figure 216 for the definitions of EL and VSIL.	

5.14.1.13.1.14 Vendor Specific Event (Event Type DEh)

<Add a table footnote to figure 234 (as modified in figure 241 in NVMe Base 1.4_NEXT 2020.07.21.docx>

Figure 234: Vendor Specific Event Format (Event Type DEh)

Bytes	Description
M-1:0	**Vendor Specific Event Descriptor 0:** Contains the first vendor specific event descriptor (refer to Figure 235). Where M is the length of this vendor specific event descriptor.
…	
EL-VSIL-1: EL-VSIL-K NOTE 1	**Vendor Specific Event Descriptor N:** Contains the last vendor specific event descriptor (refer to Figure 235) where ~~EL is the value from the Event Length field in the Persistent Event Log event header (refer to Figure 216), VSIL is the value from the Vendor Specific Information Length field in the Persistent Event Log header, and~~ K is the length of this vendor specific event descriptor (refer to Figure 235).
NOTE: 1. Refer to Figure 216 for the definitions of EL and VSIL.	

5.14.1.16.2 Sanitize Status (Log Identifier 81h)

<Modify Figure 242 (Get Log Page – Sanitize Status Log) as follows>

…

Bytes	Description
…	…
23:20	**Estimated Time For Overwrite With No-Deallocate Media Modification:** This field indicates the number of seconds required to complete an Overwrite sanitize operation and the associated additional media modification after the Overwrite sanitize operation in the background (refer to section 5.24) when: a) the No-Deallocate After Sanitize bit was set to '1' in the Sanitize command that requested the Overwrite sanitize operation; and b) the No-Deallocate Modifies Media After Sanitize field (refer to Figure 251) is set to 10b. A value of 0h indicates that the sanitize operation is expected to be completed in the background when the Sanitize command that started that operation is completed. A value of FFFFFFFFh indicates that no time period is reported.
27:24	**Estimated Time For Block Erase With No-Deallocate Media Modification:** This field indicates the number of seconds required to complete a Block Erase sanitize operation and the associated additional media modification after the Block Erase sanitize operation in the background (refer to section 5.24) when: a) the No-Deallocate After Sanitize bit was set to '1' in the Sanitize command that requested the Block Erase sanitize operation; and b) the No-Deallocate Modifies Media After Sanitize field (refer to Figure 251) is set to 10b. A value of 0h indicates that the sanitize operation is expected to be completed in the background when the Sanitize command that started that operation is completed. A value of FFFFFFFFh indicates that no time period is reported.
31:28	**Estimated Time For Crypto Erase With No-Deallocate Media Modification:** This field indicates the number of seconds required to complete a Crypto Erase sanitize operation and the associated additional media modification after the Crypto Erase sanitize operation in the background (refer to section 5.24) when: a) the No-Deallocate After Sanitize bit was set to '1' in the Sanitize command that requested the Crypto Erase sanitize operation; and b) the No-Deallocate Modifies Media After Sanitize field (refer to Figure 251) is set to 10b. A value of 0h indicates that the sanitize operation is expected to be completed in the background when the Sanitize command that started that operation is completed. A value of FFFFFFFFh indicates that no time period is reported.
511:32	Reserved

5.15.2.1 Identify Namespace data structure (CNS 00h)

…

<multiple changes to figure 249>

Figure 249: Identify – Identify Namespace Data Structure, NVM Command Set Specific

…

<modify NSZE since it is impossible to have a condition 'prior to the namespace being formatted'>

Figure 249: Identify – Identify Namespace Data Structure, NVM Command Set Specific

Bytes	O/M [1]	Description
07:00	M	**Namespace Size (NSZE):** This field indicates the total size of the namespace in logical blocks. A namespace of size *n* consists of LBA 0 through (*n* - 1). The number of logical blocks is based on the formatted LBA size. ~~This field is undefined prior to the namespace being formatted.~~
…		**…**

….

<Qualify NSFEAT when a namespace is associated with an NVM Set>

24	M	**Namespace Features (NSFEAT):** This field defines features of the namespace. …. Bit 4 (**OPTPERF**) if set to '1' indicates that the fields NPWG, NPWA, NPDG, NPDA, and NOWS are defined for this namespace and should be used by the host for I/O optimization (refer to section 8.25). If cleared to '0', then the controller does not support the fields NPWG, NPWA, NPDG, NPDA, and NOWS for this namespace. ~~• indicates that the fields NPWG, NPWA, NPDG, NPDA, and NOWS are defined for this namespace and should be used by the host for I/O optimization (refer to section XXX;~~ ~~• NOWS defined for this namespace adhere to Optimal Write Size field setting defined in NVM Sets Attributes Entry (refer to Figure 253) for the NVM Set with which this namespace is associated.~~ ~~If cleared to '0', then: the controller does not support the fields NPWG, NPWA, NPDG, NPDA, and NOWS for this namespace.~~ ~~• the controller does not support the fields NPWG, NPWA, NPDG, NPDA, and NOWS for this namespace; and~~ ~~• Optimal Write Size field in NVM Sets Attributes Entry (refer to Figure 253) for the NVM Set with which this namespace is associated should be used by the host for I/O optimization~~ …

<clarify the use of NOWS>

73:72	O	**Namespace Optimal Write Size (NOWS):** This field indicates the size in logical blocks for optimal write performance for this namespace. This is a 0's based value. The size indicated should be less than or equal to Maximum Data Transfer Size (MDTS) that is specified in units of minimum memory page size. The value of this field may change if the namespace is reformatted. The value of this field should be a multiple of Namespace Preferred Write Granularity (NPWG). If the namespace is associated with an NVM set, NOWS defined for this namespace shall be set to the Optimal Write Size field setting defined in NVM Set Attributes Entry (refer to Figure 253) for the NVM Set with which this namespace is associated. If NOWS is not supported, the Optimal Write Size field in NVM Sets Attributes Entry (refer to Figure 253) for the NVM Set with which this namespace is associated should be used by the host for I/O optimization. Refer to section 8.25 for how this field is utilized to improve performance and endurance.

…

<clarify that the firmware revision is an ASCII string>

5.15.2.2 Identify Controller data structure (CNS 01h)

Figure 251: Identify – Identify Controller Data Structure, NVM Command Set Specific

71:64	M	**Firmware Revision (FR):** Contains the currently active firmware revision, as an ASCII string, for the NVM subsystem. This is the same revision information that may be retrieved with the Get Log Page command, refer to section 5.14.1.3. ~~Refer to section 1.5 for ASCII string requirements.~~

…

<Clarify 'no deallocate' text>

Figure 251: Identify – Identify Controller Data Structure

331:328	O	**Sanitize Capabilities (SANICAP):** This field indicates attributes for sanitize operations. If the Sanitize command is supported, then this field shall be non-zero. If the Sanitize command is not supported, then this field shall be cleared to 0h. Refer to section 8.15. **Bits** | **Description** … |

Figure 251: Identify – Identify Controller Data Structure

<table>
<tr><td>31:30</td><td>No-Deallocate Modifies Media After Sanitize (NODMMAS): This field indicates if media is additionally modified by the NVMe controller after a sanitize operation successfully completes that had been started by a Sanitize command with the No-Deallocate After Sanitize bit set to ‘1’.

The work required for the associated additional media modification is included both in the estimated time for each sanitize operation and in the Sanitize Progress field (refer to Figure 242).
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00b</td><td>Additional media modification after sanitize operation completes successfully is not defined. Only controllers compliant with versions 1.3 and earlier of the specification or that have bits 2:0 of the SANICAP field cleared to 0h shall be allowed to return this value.</td></tr>
<tr><td>01b</td><td>Media is not additionally modified by the NVMe controller after sanitize operation completes successfully.</td></tr>
<tr><td>10b</td><td>Media is additionally modified by the NVMe controller after sanitize operation completes successfully. The Sanitize Operation Completed event does not occur until the additional media modification associated with this field has completed.</td></tr>
<tr><td>11b</td><td>Reserved</td></tr>
</table>
If bits 2:0 of the SANICAP field are cleared to 000b, then the controller shall clear this field to 00b.</td></tr>
<tr><td>29</td><td>No-Deallocate Inhibited (NDI): If set to ‘1’ and the No-Deallocate Response Mode bit is set to ‘1’, then the controller deallocates after the sanitize operation even if the No-Deallocate After Sanitize bit is set to ‘1’ in a Sanitize command.

If:
a) this bit is set to ‘1’;
b) the No-Deallocate After Sanitize bit is set to ‘1’ in a Sanitize command, and:
~~a) set to ‘1’;~~
1) ~~b)~~ the No-Deallocate Response Mode bit (refer to Figure 318) is cleared to ‘0’; ~~and~~ or
2) ~~c)~~ the Sanitize Config Feature (refer to section 5.21.1.23) is not supported, ~~the No-Deallocate After Sanitize bit is set to ‘1’ in a Sanitize command,~~
then the controller aborts the Sanitize command with a status of Invalid Field in Command.

If the No-Deallocate After Sanitize bit is cleared to ‘0’ in a Sanitize command, then the value of this bit has no effect on the processing of that Sanitize command.

If this bit is cleared to ‘0’, then the controller supports the No-Deallocate After Sanitize bit in a Sanitize command.

If bits 2:0 of the SANICAP field are cleared to 0h, then the controller shall clear this bit to ‘0’.</td></tr>
<tr><td>28:03</td><td>Reserved</td></tr>
</table>

5.15.2.5 NVM Set List (CNS 04h)

<add clarifying text>

Figure 254 defines an NVM Set List. The data structure is an ordered list of NVM Set Attribute Entry data structures, sorted by NVM Set Identifier, starting with the first NVM Set Identifier supported by the NVM subsystem that is equal to or greater than the NVM Set Identifier indicated in CDW11.NVMSETID. The NVM Set List describes the attributes for each NVM Set in the list based on the NVM Set Attributes Entry in Figure 254.

5.21.1 Feature Specific Information

<clarify unsaved feature settings after a controller level reset>

Figure 275 defines the Features that may be configured with a Set Features command and retrieved with a Get Features command. Figure 276 defines Features that are specific to the NVM Command Set. Refer to section 7.1 for mandatory, optional, and prohibited features for the various controller types. Some Features utilize a memory buffer to configure or return attributes for a Feature, whereas others only utilize a dword in the command or completion queue entry. ~~Feature values that are not persistent across power cycles and resets are restored to their default values as part of a controller reset operation.~~ If a Feature is not persistent across power cycles and resets, then the current value of that Feature shall be set to the default value of that Feature as part of a Controller Level Reset. For more information on Features, including default value definitions, saveable value definitions, and current value definitions, refer to section 7.8.

…

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

<clarify timestamp use>

Timestamp values should not be used for security applications. Other application ~~The~~ use of the Timestamp is outside the scope of this specification.

…

5.21.1.23 Sanitize Config (Feature Identifier 17h), (Optional)

<Add a hyphen to “No Deallocate After Sanitize” in figure 334>

Figure 334: Sanitize Config – Command Dword 11

Bits	Description
31:01	Reserved

Figure 334: Sanitize Config – Command Dword 11

Bits	Description
00	**No-Deallocate Response Mode (NODRM):** If the No-Deallocate Inhibited bit in the Sanitize Capabilities field of the Identify Controller data structure (refer to Figure 251) is set to '1', then the NODRM bit defines the response of the controller to a Sanitize command processed with the No-Deallocate ~~No-Deallocate~~ After Sanitize bit (refer to Figure 334) set to '1'. If the NODRM bit is set to '1' (i.e., No-Deallocate Warning Response Mode), then the controller shall process such Sanitize commands, and if the resulting sanitize operation is completed successfully, then bits 2:0 of the Sanitize Status field in the Sanitize Status log page shall be set to 100b (refer to Figure 242). If the NODRM bit is cleared to '0' (i.e., No-Deallocate Error Response Mode), then the controller shall abort such Sanitize commands with a status of Invalid Field in Command. If the No-Deallocate Inhibited bit in the Sanitize Capabilities field of the Identify Controller data structure (refer to Figure 251) is cleared to '0', then this bit has no effect.

5.23 Format NVM command – NVM Command Set Specific

<correct the name of the Format NVM command>

…

For a Format NVM command with the NSID field set to FFFFFFFFh that specifies secure erase:

a) if bit 1 is set to '1' in the FNA field (refer to Figure 251) and there are no namespaces in the NVM subsystem, then that Format NVM command shall complete without error; and

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

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

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

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

…

< clarify the intent of User Data Erase>

Figure 332: Format NVM – Command Dword 10

<table>
<tr><th>Bits</th><th colspan="3">Description</th></tr>
<tr><td>31:12</td><td colspan="3">Reserved</td></tr>
<tr><td rowspan="4">11:09</td><td colspan="3">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.).</td></tr>
<tr><td></td><td>Value</td><td>Definition</td></tr>
<tr><td></td><td>000b</td><td>No secure erase operation requested</td></tr>
<tr><td></td><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~~. If a User Data Erase is requested and all affected user data is encrypted, then the controller is allowed to use a cryptographic erase to perform the requested User Data Erase.</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td colspan="3"></td></tr>
</table>

5.24 Sanitize command – NVM Command Set Specific

…

<Add a hyphen to “No Deallocate After Sanitize” in figure 334>

Figure 334: Sanitize – Command Dword 10

Bits	Description
09	**No-Deallocate** ~~**No Deallocate**~~ **After Sanitize:** If set to ‘1’ and the No-Deallocate Inhibited bit (refer to Figure 251) is cleared to ‘0’, then the controller shall not deallocate any logical blocks as a result of successfully completing the sanitize operation. If: a) cleared to ‘0’; or b) set to ‘1’ and the No-Deallocate Inhibited bit is set to ‘1’, then the controller should deallocate logical blocks as a result of successfully completing the sanitize operation. This bit shall be ignored if the Sanitize Action field is set to 001b (i.e., Exit Failure Mode).

6.1.5 NSID and Namespace Relationships

<Modify Figure 351: NSID Types and Relationship to Namespace>

Valid NSID Type	NSID relationship to namespace	Reference
Unallocated	Does not refer to any namespace that exists in the NVM subsystem	6.1.3
Allocated	Refers to a namespace that exists in the NVM subsystem	6.1.3
Inactive	Does not refer to a namespace that is attached to the controller[1]	6.1.4
Active	Refers to a namespace that is attached to the ~~this~~ controller	6.1.4
NOTES: 1. If allocated, refers to a namespace that is not attached to the controller. If unallocated, does not refer to any namespace.		

6.12 Reservation Release command

<change ‘registration’ to ‘reservation’ >

Figure 389: Reservation Release – Command Dword 10

Bits	Description
31:16	Reserved
15:08	**Reservation Type (RTYPE):** If the Reservation Release Action field is cleared to 000b (i.e., Release), then this field specifies the type of reservation that is being released. The reservation type in this field shall match the current reservation type. If the reservation type in this field does not match the current reservation type, then the controller should return an error of Invalid Field In Command. This field is defined in Figure 384.
07:04	Reserved
03	**Ignore Existing Key (IEKEY)**: If this bit is set to a ‘1’, the controller shall return an error of Invalid Field in Command. If this bit is cleared to ‘0’, then the Current Reservation Key is checked.

Figure 389: Reservation Release – Command Dword 10

<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>02:00</td><td>Reservation Release Action (RRELA): This field specifies the ~~registration~~ reservation action that is performed by the command.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>000b</td><td>Release</td></tr>
<tr><td>001b</td><td>Clear</td></tr>
<tr><td>010b to 111b</td><td>Reserved</td></tr>
</table></td></tr>
</table>

7.1.2 Administrative Controller

<change the figure title>

Figure 432: Administrative Controller – ~~Controller~~ Log Page Support

7.6.1 Initialization

<delete a paragraph at the bottom of section 7.6.1>

…

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

For exit of the D3 power state, the initialization steps outlined should be followed. ~~In this case, the number of I/O Submission Queues and I/O Completion Queues shall not change, thus step 7 of the initialization sequence is optional.~~

7.8 Feature Values

<Modify 7.8 Feature Values>

Each Feature has supported capabilities (refer to Figure 188), which are discovered by using the ~~'supported capabilities'~~ Supported Capabilities value in the Select field in Get Features (refer to Figure 187).

7.12 Keep Alive

<modify 7.12 as noted>

…

The Keep Alive timer is active if:

- CC.EN is set to '1';
- CSTS.RDY is set to '1';
- CC.SHN is cleared to '00b';
- CSTS.SHST is cleared to '00b'; and
- the Keep Alive Timer feature has been enabled with a KATO field (refer to section 5.21.1.15 or the Fabric Connect command in the NVMe over Fabrics specification) set to a non-zero value,

otherwise, the Keep Alive timer is inactive and a Keep Alive Timeout shall not occur. Activating an inactive Keep Alive timer (e.g., enabling a controller with the Keep Alive feature in use, enabling a controller that supports NVMe-oF where the Connect command specified a non-zero Keep Alive Timeout (refer to the NVMe over Fabrics specification)) shall initialize the Keep Alive timer to the Keep Alive Timeout (KATO) value.

8.3.1 PRACT bit

~~8.3.1 The PRACT Bit~~

<change the title>

8.3.1.5 Control of Protection Information Checking – PRCHK field (PRCHK)

<renumber as 8.3.2 because the PRCHK (field) is disjoint from the PRACT (bit)>

~~8.3.1.5~~ 8.3.2 Control of Protection Information Checking ~~– PRCHK field~~ (PRCHK)

8.4 Power Management

<Modify 8.4 Power Management>

…

Associated with each power state is a Power State Descriptor in the Identify Controller data structure (refer to Figure 252). The descriptors for all implemented power states may be viewed as forming a table as shown in the example in Figure 460 for a controller with seven implemented power states. Note that Figure 460 is illustrative and does not include all fields in the power state descriptor. The Maximum Power (MP) field indicates the sustained maximum power that may be consumed in that state, where power measurement methods are outside the scope of this specification ~~(e.g., refer to the appropriate form factor specification for power measurement methodologies for that form factor)~~. The controller may employ autonomous power management techniques to reduce power consumption below this level, but under no circumstances is power allowed to exceed this level except for non-operational power states as described in section 8.4.1.

…

8.12 Namespace Management (Optional)

<Modify a portion of section 8.12>

To create a namespace, host software performs the following actions:

1. Host software requests the Identify Namespace data structure that specifies common namespace capabilities (i.e., using an Identify command with the NSID field set to FFFFFFFFh and the CNS field cleared to 0h);
2. If the controller supports reporting of Namespace Granularity, host software optionally requests the Namespace Granularity List defined in Figure 259 (Identify command with CNS set to 16h).
3. Host software creates the data structure defined in Figure 269. Host software sets the host software specified fields defined in Figure 266 (taking into account the common namespace capabilities);
4. Host software issues the Namespace Management command specifying the Create operation and the data structure. On successful completion of the command, the Namespace Identifier of the new namespace is returned in Dword 0 of the completion queue entry. At this point, the new namespace is not attached to any controller; and
5. Host software requests the Identify Namespace data structure for the new namespace to determine all attributes of the namespace.

To attach a namespace, host software performs the following actions:

1. Host software issues the Namespace Attachment command specifying the Controller Attach operation to attach the ~~new~~ specified namespace to one or more controllers; and
2. If Namespace Attribute Notices are enabled, the controller(s) newly attached to the namespace report a Namespace Attribute Changed asynchronous event to the host.

To detach a namespace, host software performs the following actions:

1. Host software issues the Namespace Attachment command specifying the Controller Detach operation to detach the specified namespace from one or more controllers; and
2. If Namespace Attribute Notices are enabled, the controllers that were detached from the namespace report a Namespace Attribute Changed asynchronous event to the host.

To delete a namespace, host software performs the following actions:

1. Host software should detach the namespace from all controllers;
2. Host software issues the Namespace Management command specifying the Delete operation for the specified namespace. On successful completion of the command, the namespace has been deleted; and
3. If Namespace Attribute Notices are enabled, any controller(s) not processing the Namespace Management command that was attached to the namespace reports a Namespace Attribute Changed asynchronous event to the host.

8.15 Sanitize Operations (Optional)

<Add a hyphen to “No Deallocate After Sanitize” in section 8.15>

…..

To start a sanitize operation, the host submits a Sanitize command specifying one of the sanitize operation types (i.e., Block Erase, Overwrite, or Crypto Erase). The host sets command parameters, including the Allow Unrestricted Sanitize Exit bit and the No-Deallocate ~~No Deallocate~~ After Sanitize bit. After validating the Sanitize command parameters, the controller starts the sanitize operation in the background, updates the Sanitize Status log page and then completes the Sanitize command with Successful Completion status. If the sanitize operation is to be followed by an associated additional media modification operation (refer to NODMMAS in Figure 251), then the associated additional media modification operation shall be completed before the controller reports sanitize operation complete. If a Sanitize command is completed with any status other than Successful Completion, then the controller shall not start the sanitize operation and shall not update the Sanitize Status log page. The controller ignores Critical Warning(s) in the SMART / Health Information log page (e.g., read only mode) and attempts to complete the sanitize operation requested. While a sanitize operation is in progress, all controllers shall abort any commands not listed in Figure 486 with a status of Sanitize In Progress (refer to section 8.15.1).

The user data values that result from a successful sanitize operation are specified in Figure 485. If the controller deallocates user data after successful completion of a sanitize operation, then values read from deallocated logical blocks are described in section 6.7.1.1. The host may specify that sanitized logical blocks not be deallocated by setting the No-Deallocate ~~No Deallocate~~ After Sanitize bit to ‘1’ in the Sanitize command.

…

The Sanitize Config Feature Identifier (refer to section 5.21.1.23) contains the No-Deallocate Response Mode bit that specifies the response of the controller to a Sanitize command processed with the No-Deallocate ~~No Deallocate~~ After Sanitize bit (refer to Figure 334) set to ‘1’ if the No-Deallocate Inhibited bit in the Sanitize Capabilities field of the Identify Controller data structure (refer to Figure 251) is set to ‘1’. In the No-Deallocate Error Response Mode, the controller aborts such Sanitize commands with a status of Invalid Field in Command. In the No-Deallocate Warning Response Mode, the controller processes such Sanitize commands, and if a resulting sanitize operation is completed successfully, then bits 2:0 of the Sanitize Status field in the Sanitize Status log page are set to 100b (refer to Figure 242).

8.20.1 Asymmetric Namespace Access Reporting Overview

...

<add the word ‘the’>

Namespaces attached to a controller that supports Asymmetric Namespace Access Reporting shall:

- be members of an ANA Group; and
- supply a valid ANA Group Identifier in the ANA Group Identifier (ANAGRPID) field in the Identify Namespace data structure (refer to Figure 249).

8.20.2 ANA Groups

<change ‘the’ to ‘this’>

Namespaces that are members of the same ANA Group perform identical asymmetric namespace access state transitions. The ANA Group maintains the same asymmetric namespace access state for all namespaces that are members of that ANA Group (i.e., a change in the asymmetric namespace access state of one namespace only occurs as part of a change in the asymmetric namespace access state of all namespaces that are members of that ANA Group). The method for assigning namespaces to ANA Groups is outside the scope of this ~~the~~ specification.

…

8.20.4 Asymmetric Namespace Access States Command Processing Effects

<change ‘this’ to ‘the’>

Figure 498: ANA effects on Command Processing

Command	ANA State	Effects on command processing
…	…	…
Set Features	ANA Inaccessible	…
	ANA Change	…
	ANA Persistent Loss	The ~~This~~ command shall fail with a status code of Asymmetric Access Persistent Loss (refer to section 8.20.3.4).
…		

…

8.21.3 Host ANA Persistent Loss Operation

<clarify what ‘supports Namespace Management’ means.>

If the ANA Log page reports an ANA state of ANA Persistent Loss State for an ANA Group or a command returns a status of Asymmetric Namespace Access Persistent Loss, then the host should not use that controller to send commands to any namespace in that ANA Group, and select a different controller for sending commands to any namespace in that ANA Group. If the controller supports the Namespace Management capability (refer to section 8.12), then the namespaces in an ANA Group reporting this state should be detached.

8.25.1 Improved I/O examples (non-normative)

…

<remove a SHALL (at the bottom of) an informative section>

If NVM Sets are supported as described in Figure 251, the value in the NOWS field for the namespace indicates the value the host should use to achieve optimal performance. ~~If NVM Sets are supported, NOWS setting for the namespace namespace shall adhere to NVM Sets Optimal Write Size setting for the NVM Set which the namespace is associated with.~~ If an NVM Set does not specify an Optimal Write Size, the host should ~~consult~~ use the value in the NOWS ~~setting~~ field for the namespace for I/O optimization purposes. Similarly, if NOWS is not defined for a namespace, the host should ~~consult~~ use the value in the Optimal Write Size ~~setting~~ field for the NVM Set associated with that namespace to achieve optimal performance.

…