LEGAL NOTICE:

This erratum to the NVM Express revision 1.2 specification is proprietary to the NVM Express, Inc. (also referred to as "Company") and/or its successors and assigns.

NOTICE TO USERS WHO ARE NVM EXPRESS, INC. MEMBERS: Members of NVM Express, Inc. have the right to use and implement this erratum to the NVM Express revision 1.2 specification subject, however, to the Member's continued compliance with the Company's Intellectual Property Policy and Bylaws and the Member's Participation Agreement.

NOTICE TO NON-MEMBERS OF NVM EXPRESS, INC.: If you are not a Member of NVM Express, Inc. and you have obtained a copy of this document, you only have a right to review this document or make reference to or cite this document. Any such references or citations to this document must acknowledge NVM Express, Inc. copyright ownership of this document. The proper copyright citation or reference is as follows: **"© 2007 - 2016 NVM Express, Inc. ALL RIGHTS RESERVED."** When making any such citations or references to this document you are not permitted to revise, alter, modify, make any derivatives of, or otherwise amend the referenced portion of this document in any way without the prior express written permission of NVM Express, Inc. Nothing contained in this document shall be deemed as granting you any kind of license to implement or use this document or the specification described therein, or any of its contents, either expressly or impliedly, or to any intellectual property owned or controlled by NVM Express, Inc., including, without limitation, any trademarks of NVM Express, Inc.

LEGAL DISCLAIMER:

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

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

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

NVM Express™ Technical Errata

Errata ID	**008**
Revision Date	**4/19/2016**
Affected Spec Ver.	**NVM Express™ 1.2a**
Corrected Spec Ver.	

Errata Author(s)

Name	Company
John Carroll, Matthew Wilcox	Intel
Dave Landsman	SanDisk
Judy Brock	Samsung
Jim Hatfield	Seagate
David Black	EMC
Jason Gao, Nadesan Narenthiran	HGST
Peter Onufryk	PMC
Fred Knight	NetApp

Errata Overview

A substantial re-write of the Protection Information section is complete to make the associated material easier to understand.

It is clarified that a controller may re-use NGUID and EUI64 values when the original namespace has been deleted.

The error condition to return when an invalid starting NSID is specified for returning a list of NSIDs is returned.

Clarifications are made to avoid implying that a controller ID of 0h is invalid.

A clarification is made to be clear that the Phase Bit shall be part of the last write done to host memory for a Completion Queue Entry.

The RTD3 resume and entry latency values may not be supported by compliant revision 1.2+ devices.

Revision History

Revision Date	Change Description
12/09/2015	Initial draft
12/16/2015	APST clarification; 4.1 wording changes; added temperature threshold clarification
12/21/2015	Added the rest of the protection information.
1/7/2016	Updated section 4.6 wording, moved RTD3 changes from Fabrics base diff, added 1-based convention, removed self-test changes from this ECN (since they are for a 1.3 feature), added clarification for optional log pages
1/28/2016	Added a few controller identifier = 0h clarifications.
2/3/2016	Minor updates, including updating the overview. Pushing additional changes to ECN 009.
4/19/2016	Ratified.

Description of Specification Changes

Modify a portion of Section 7.9 (Unique Identifier) as shown below:

The Identify Namespace data structure contains the IEEE Extended Unique Identifier (EUI64) and the Namespace Globally Unique Identifier (NGUID) fields. EUI64 is an 8-byte EUI-64 identifier and NGUID is a 16-byte identifier based on EUI-64. When creating a namespace, the controller specifies a globally unique value in the EUI64 or NGUID field (the controller may optionally specify a globally unique value in both fields). In cases where the 64-bit EUI64 field is unable to ensure a globally unique namespace identifier, the EUI64 field shall be cleared to 0h. When not implemented, these fields contain a value of 0h. A controller may reuse a non-zero NGUID or EUI64 value for a new namespace after the original namespace using the value has been deleted.

Modify a portion of Figure 86 (Identify – Data Structure Returned) as shown below:

CNS Value	Definition
. . .	. . .
Namespace Management	
10h	A list of up to 1024 namespace IDs is returned to the host containing allocated NSIDs with a namespace identifier greater than the value specified in the Namespace Identifier (CDW1.NSID) field. The controller should abort the command with status code Invalid Namespace or Format if CDW1.NSID is set to FFFFFFFEh or FFFFFFFFh. Note that CDW1.NSID may be cleared to 0h to retrieve a Namespace List including the namespace starting with NSID of 1h.

Modify a portion of Figure 1 (Asynchronous Event Information – Notice) as shown below:

Value	Description
0h	**Namespace Attribute Changed:** The Identify Namespace data structure for one or more namespaces, as well as the Namespace List returned when the Identify command is issued with the CNS field set to 02h, ~~has~~ have changed. Host software may use this event as an indication that it should read the Identify Namespace data structures for each namespace to determine what has changed. Alternatively, host software may request the Changed Namespace List (Log Identifier 04h) to determine which namespaces in this controller have changed Identify Namespace information since the last time the log page was read. A controller shall not send this event when Namespace Utilization has changed, as this is a frequent event that does not require action by the host. A controller shall only send this event for changes to the Format Progress Indicator field when bits 6:0 of that field transition from a non-zero value to zero or from a zero value to a non-zero value.

Modify section 4.9 as shown below:

A Controller List, defined in Figure 37, is an ordered list of ascending controller IDs. The controller identifier is defined in bytes 79:78 of the Identify data structure in Figure 90. Unused entries are zero filled.

Figure 37: Controller List Format

Bytes	Description
1:0	**Number of Identifiers:** This field contains the number of controller entries in the list. There may be up to 2047 identifiers in the list. A value of 0 indicates there are no controllers in the list.
3:2	**Identifier 0:** This field contains the NVM subsystem unique controller identifier for the first controller in the list if present. ~~or 0h if the list is empty (i.e. there are no controllers in the list).~~
5:4	**Identifier 1:** This field contains the NVM subsystem unique controller identifier for the second controller in the list if present. ~~or 0h if the list contains fewer than two entries.~~
…	…
(N*2+3): (N*2+2)	**Identifier N:** This field contains the NVM subsystem unique controller identifier for the N+1 controller in the list if present. ~~or 0h if the list contains fewer than N+1 entries.~~

Modify Figure 89 as shown below:

Figure 89: Identify – Command Dword 10

Bit	Description
31:16	**Controller Identifier (CNTID):** This field specifies the controller identifier used as part of some Identify operations. If the field is not used as part of the Identify operation, then host software shall clear this field to 0h for backwards compatibility (0h is a valid controller identifier). Controllers that support Namespace Management shall support this field.
15:08	Reserved
07:00	**Controller or Namespace Structure (CNS):** This field specifies the information to be returned to the host. Refer to Figure 86.

Modify a portion of* section 4.6 (Completion Queue Entry) *as shown below:

An entry in the Completion Queue is at least 16 bytes in size. Figure 25 describes the layout of the first 16 bytes of the Completion Queue Entry data structure. The contents of Dword 0 are ~~is~~ command specific. If a

command uses Dword 0, then the definition of this Dword is contained within the associated command definition. If a command does not use Dword 0, then the field is reserved. Dword 1 is reserved. Dword 2 is defined in Figure 26 and Dword 3 is defined in Figure 27. Any additional I/O Command Set defined in the future may use an alternate Completion Queue entry size or format.

If a Completion Queue Entry is constructed via multiple writes, the Phase Tag bit shall be updated in the last write of that Completion Queue Entry.

Modify section 8.3 (End to End Data Protection) as shown below:

To provide robust data protection from the application to the NVM media and back to the application itself, end-to-end data protection may be used. If this optional mechanism is enabled, then additional protection information (e.g. CRC) is added to the logical block that may be evaluated by the controller and/or host software to determine the integrity of the logical block. This additional protection information, if present, is either the first eight bytes of metadata or the last eight bytes of metadata, based on the format of the namespace. For metadata formats with more than eight bytes, if the protection information is contained within the first eight bytes of metadata, then the CRC does not cover any metadata bytes. For metadata formats with more than eight bytes, if the protection information is contained within the last eight bytes of metadata, then the CRC covers all metadata bytes up to but excluding these last eight bytes. As described in section 8.2, metadata, and hence this protection information may be configured to be contiguous with the logical block data or stored in a separate buffer.

The most commonly used data protection mechanisms in Enterprise implementations are SCSI Protection Information, commonly known as Data Integrity Field (DIF), and the Data Integrity Extension (DIX). The primary difference between these two mechanisms is the location of the protection information. In ~~SCSI Protection Information~~ 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 ~~SCSI Protection Information~~ DIF and DIX. ~~SCSI Protection Information~~ DIF functionality is achieved by configuring the metadata to be contiguous with logical block data (as shown in Figure 211), while DIX functionality is achieved by configuring the metadata and data to be in separate buffers (as shown in Figure 212).

NVM Express supports the same end-to-end protection types as DIF. The type of end-to-end data protection (Type 1, Type 2, or Type 3) is selected when a namespace is formatted and is reported in the Identify Namespace data structure.

The Protection Information format is shown in Figure 213 and is contained in the metadata associated with each logical block. The Guard field contains a CRC-16 computed over the logical block data. In addition to a CRC-16, DIX also specifies an optional IP checksum that is not supported by NVM Express. The Application Tag is an opaque data field not interpreted by the controller and that may be used to disable checking of protection information. The Reference Tag associates logical block data with an address and protects against misdirected or out-of-order logical block transfer. Like the Application Tag, the Reference Tag may also be used to disable checking of protection information.

Figure 213: Protection Information Format

8.3.1 The PRACT Bit

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

8.3.1.1 Protection Information and Write Commands

Figure 214 ~~illustrates~~ 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 ~~any~~ ~~optional~~ metadata is transferred from the host to the NVM with no protection information related processing by the controller.

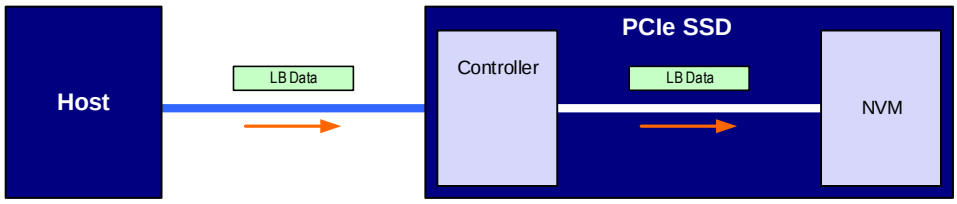
If the namespace ~~was~~ is formatted with protection information and the PRACT bit is cleared to '0', then logical block data and metadata, which contains the ~~containing~~ protection information and may contain additional metadata, are transferred from the host buffer to NVM (i.e., the metadata field remains the same size in the NVM and the host buffer). As the logical block data and metadata passes through the controller, the protection information is checked. If a protection information check error is detected, the command completes with the status code of the error detected (i.e., End-to-end Guard Check, End-to-end Application Tag Check or End-to-end Reference Tag Check).

If the namespace ~~was~~ is formatted with protection information and the PRACT bit is set to '1', then: ~~logical block data is transferred from the host to the controller.~~

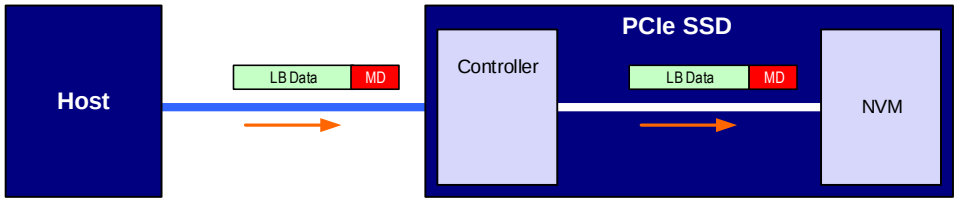
1. If the namespace is formatted with Metadata Size equal to 8 (refer to Figure 93), then the logical block data is transferred from the host buffer to the controller. As the logical block data passes through the controller, the ~~The~~ controller ~~inserts~~ generates and appends protection information to the end of the logical block data, and the logical block data and ~~metadata containing~~ protection information are written to NVM (i.e., the metadata is not resident within the host buffer).
2. If the namespace is formatted with Metadata Size greater than 8, then the logical block data and the metadata are transferred from the host buffer to the controller. As the metadata passes through the controller, the controller overwrites the protection information portion of the metadata. The logical block data and metadata are written to the NVM (i.e., the metadata field remains the same size in the NVM and the host buffer). The location of the protection information within the metadata is configured when the namespace is formatted (refer to the DPS field in Figure 90).

~~When there is additional metadata, the protection information may be transferred as the first eight bytes of metadata or the last eight bytes of metadata. The location of the protection information is configured when the namespace is formatted. In this case, the protection information replaces the first eight or last eight bytes of metadata (i.e., the metadata field remains the same size in NVM as that in the host).~~

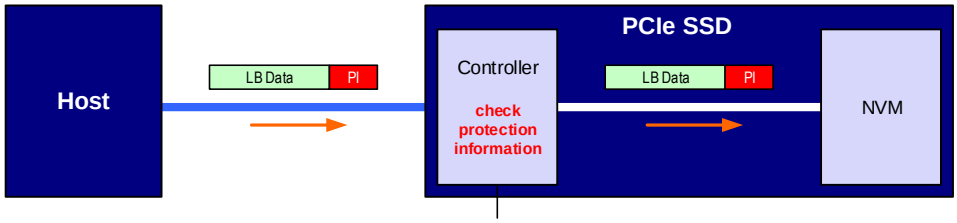
Figure 214: Write Command Protection Information Processing



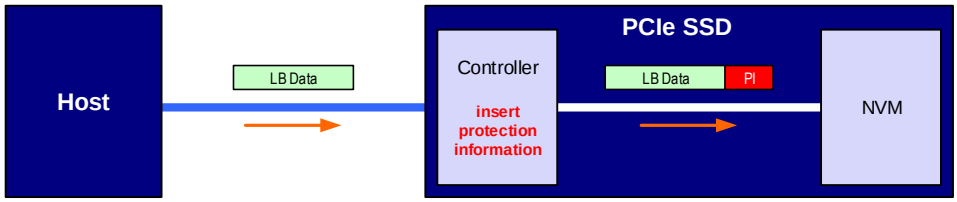
(a) No Protection Information



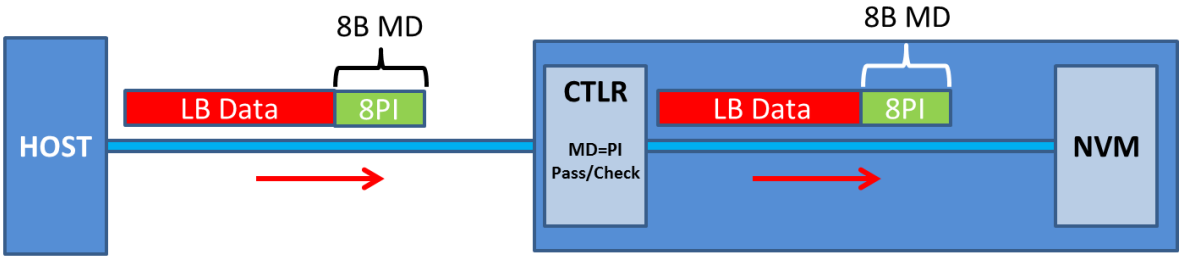
(b) No Protection Information with metadata



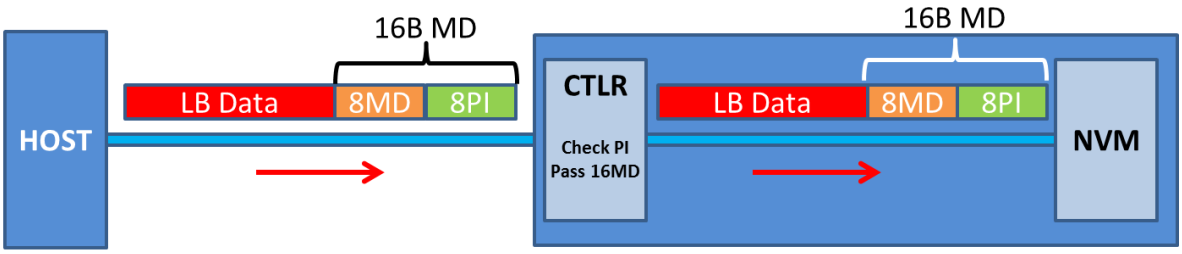
(c) Protection Information with PRACT bit cleared to '0' (i.e., pass)



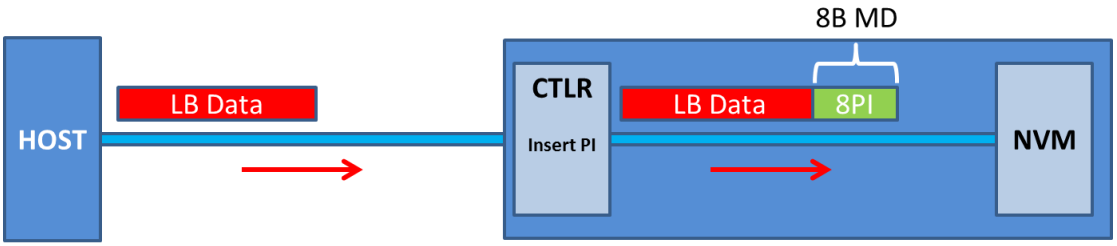
(d) Protection Information with PRACT bit set to '1' (i.e., insert)



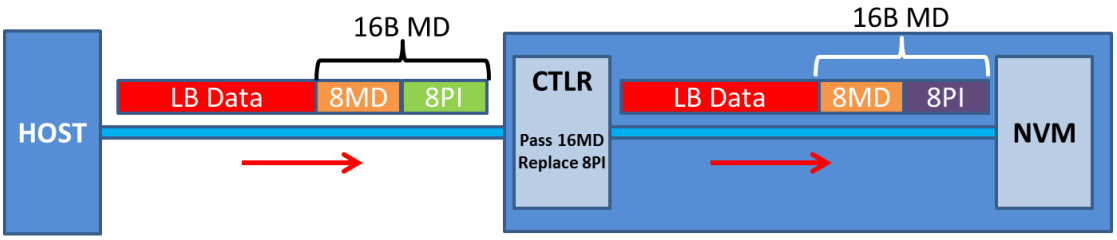
a) MD=8, PI, PRACT=0: Metadata remains same size in NVM and host buffer



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



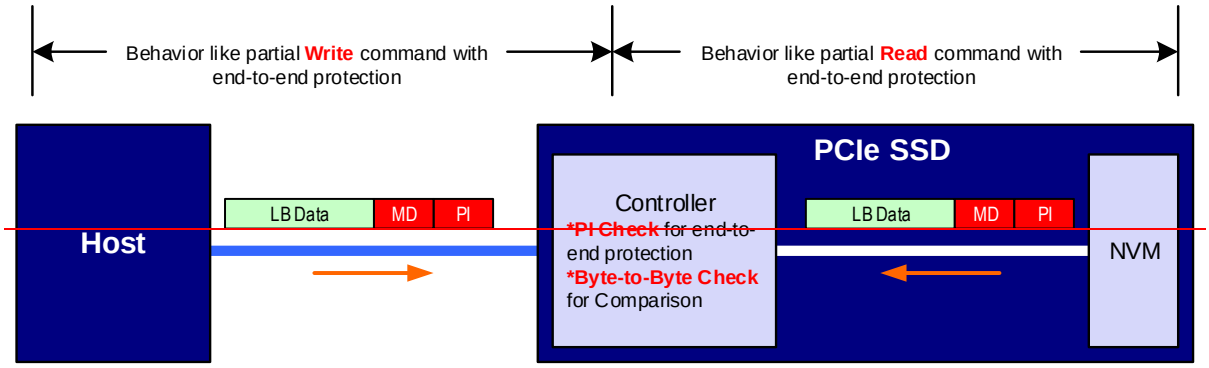
d) MD>8 (e.g. 16), PI, PRACT=1: Metadata remains same size in NVM and host buffer

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

8.3.1.2 The PRACT Bit and Read Commands

~~Figure 2 illustrates the protection information processing that may occur as a side effect of Compare command processing. Compare command processing parallels both Write and Read commands. The controller checks the protection information contained in the command and the protection information read from the NVM.~~

~~**Figure 2: Protection Information Processing for Compare**~~



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

Figure 216 provides some examples of ~~illustrates~~ the protection information processing that may occur as a side effect of Read command processing. ~~The processing parallels Write command processing with the exception that logical block data flows in the opposite direction. When the PRACT bit is cleared to '0' and the namespace was formatted with protection information, logical block data and metadata are transferred from NVM to the host and checked by the controller. When the PRACT bit is set to '1' and the namespace was formatted with protection information, logical block data and metadata are transferred from the NVM to the controller. The controller checks the protection information and then removes it from the metadata before passing the LBA to the host. If the namespace format contains metadata beyond the protection information, then the protection information is not stripped regardless of the state of the PRACT bit (i.e., the metadata field remains the same size in the host as that in NVM).~~

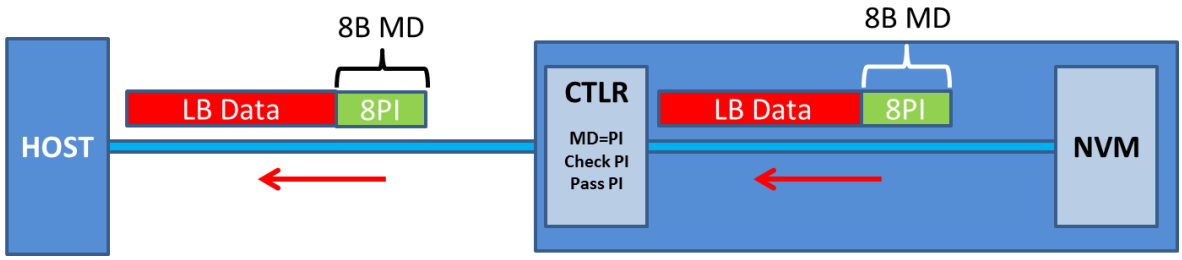
If the namespace is formatted with protection information and the PRACT bit is cleared to '0', then the logical block data and metadata, which in this case contains the protection information and possibly additional host metadata, is transferred by the controller from the NVM to the host buffer (i.e., the metadata field remains the same size in the NVM and the host buffer). As the logical block data and metadata pass through the controller, the protection information within the metadata is checked. If a protection information check error is detected, the command completes with the status code of the error detected (i.e., End-to-end Guard Check, End-to-end Application Tag Check or End-to-end Reference Tag Check).

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

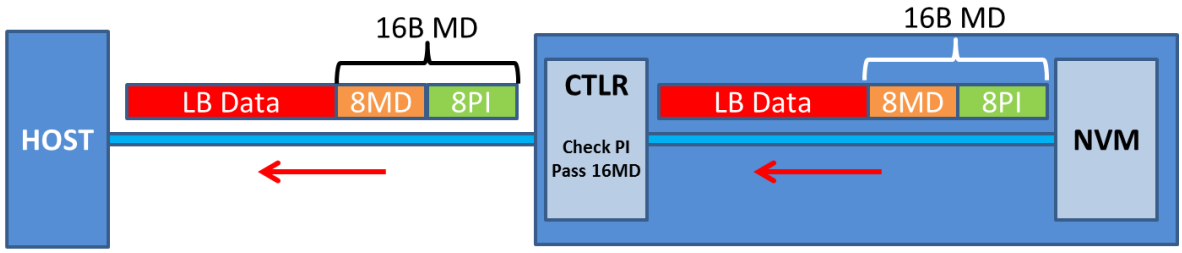
a) if the namespace is formatted with Metadata Size equal to 8 (see Figure 93), the logical block data and metadata (which in this case is, by definition, the protection information)~~,~~ is read from the NVM by the controller. As the logical block and metadata pass through the controller, the protection information is checked. If a protection information check error is detected, the command completes with the status code of the error detected (i.e., End-to-end Guard Check, End-to-end Application Tag Check or End-to-end Reference Tag Check). After processing the protection information, the controller strips it and returns the logical block data to the host (i.e., the metadata is not resident within the host buffer);

b) if the namespace is formatted with Metadata Size greater than 8, the logical block data and the metadata, which in this case contains the protection information and additional host formatted metadata, is read from the NVM by the controller. As the logical block and metadata pass through the controller, the protection information embedded within the metadata is checked. If a protection information check error is detected, the command completes with the status code of the error detected (i.e., End-to-end Guard Check, End-to-end Application Tag Check or End-to-end Reference Tag Check). After processing the protection information, the controller passes the logical block data and metadata, with the embedded protection information unchanged, to the host (i.e., the metadata field remains the same size in the NVM as within the host buffer).

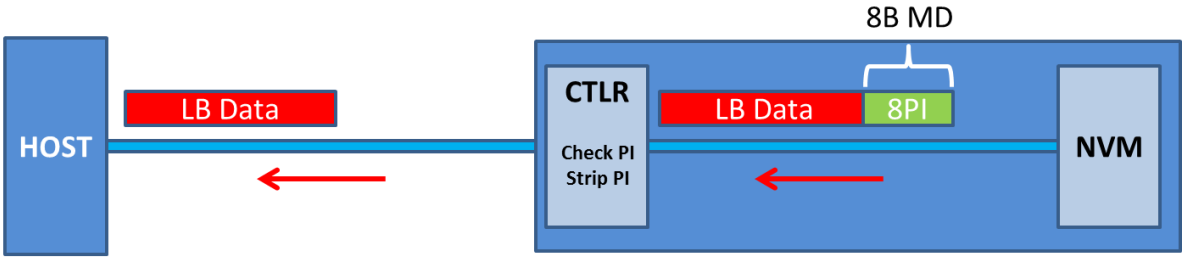
Figure 216: Read Command Protection Information Processing



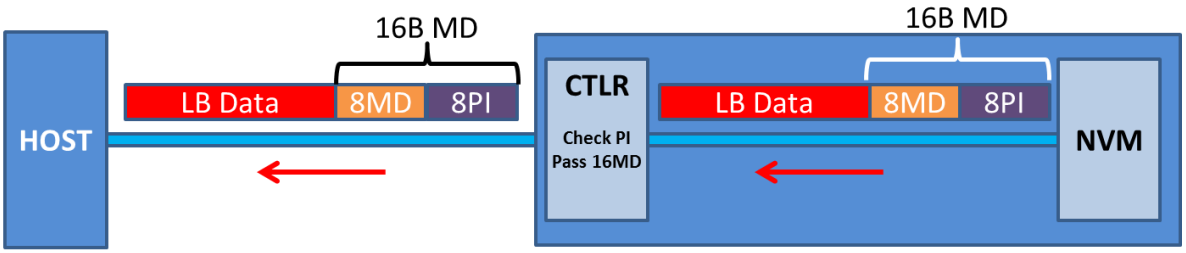
a) MD=8, PI, PRACT=0: Metadata remains same size in NVM and host buffer



b) MD>8 (e.g., 16), PI, PRACT=0: Metadata remains same size in NVM and host buffer



c) MD=8; PI, PRACT=1: Metadata not resident in host buffer



d) MD>8 (e.g. 16), PI, PRACT=1: Metadata remains same size in NVM and host buffer

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

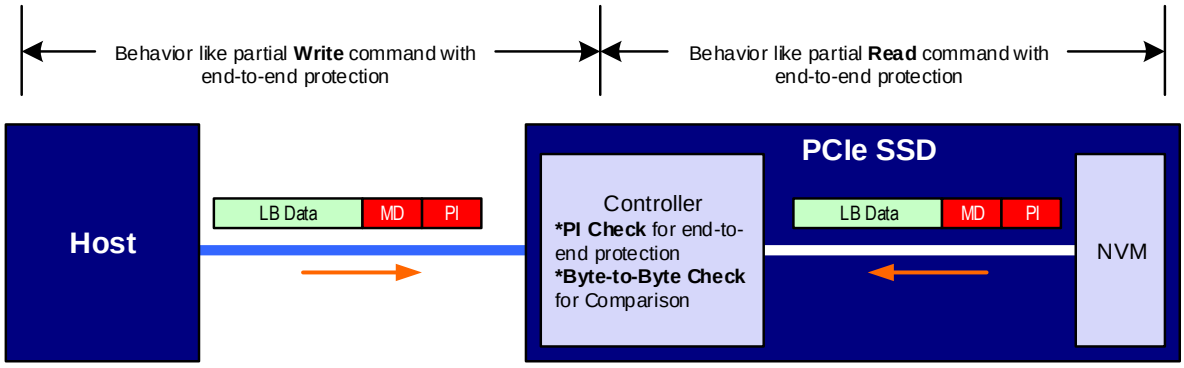
8.3.1.3 Protection Information for Fused Operations

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

8.3.1.4 Protection Checking with the Compare command

Figure 215 illustrates the protection information processing that may occur as a side effect of Compare command processing. Compare command processing parallels both Write and Read commands. The controller checks the protection information contained in the command and the protection information read from the NVM.

Figure 215: 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 bit 2 of the Protection Information Check (PRCHK) field of the command is set to '1', then the controller compares the protection information Guard field to the CRC-16 computed over the logical block data. If bit 1 of the PRCHK field is set to '1', then the controller compares unmasked bits in the protection information Application Tag field to the Logical Block Application Tag (LBAT) field in the command. A bit in the protection information Application Tag field is masked if the corresponding bit is cleared to '0' in the Logical Block Application Tag Mask (LBATM) field of the command.

For Type 1 protection, if bit 0 of the PRCHK field is set to '1', then the controller compares the protection information Reference Tag field to the computed reference tag. The value of the computed reference tag for the first LBA of the command is the value contained in the Initial Logical Block Reference Tag (ILBRT) or Expected Initial Logical Block Reference Tag (EILBRT) field in the command. If the namespace is formatted for Type 1 or Type 2 protection, the computed reference tag is incremented for each subsequent logical block. If the namespace is formatted for Type 3 protection, the reference tag for each subsequent logic block remains the same as the initial reference tag. Unlike SCSI Protection Information Type 1 protection which implicitly uses the least significant four bytes of the LBA, the controller always uses the ILBRT or EILBRT field and requires host software to initialize the ILBRT or EILBRT field to the least significant four bytes of the LBA when Type 1 protection is used. In Type 1 protection, the controller should check the ILBRT or EILBRT field; if there is any miscompare, the command completes with an error of Invalid Protection Information. If the ILBRT or EILBRT field does not match the least significant four bytes of the LBA, then the controller completes the command with an Invalid Protection Information status code.

For Type 2 protection, if bit 0 of the PRCHK field is set to '1', then the controller compares the protection information Reference Tag field from each logical block to the computed reference tag. The computed reference tag is incremented for each subsequent logical block. The value of the computed reference tag for the first LBA of the command is the value contained in the ILBRT or EILBRT field in the command. Host software may set the ILBRT and EILBRT fields to any value.

For Type 3 protection, if bit 0 of the PRCHK field is set to '1', then the command may be aborted with status Invalid Field in Command. The controller may ignore the ILBRT and EILBRT fields when Type 3 protection is used because the computed reference tag remains unchanged.

Protection checking may be disabled as a side effect of the value of the protection information Application Tag and Reference Tag fields regardless of the state of the PRCHK field in the command. If the namespace is formatted for Type 1 or Type 2 protection, then all protection information checks are disabled regardless of the state of the PRCHK field when the protection information Application Tag has a value of FFFFh. If the namespace is formatted for Type 3 protection, then all protection information checks are disabled regardless of the state of the PRCHK field when the protection information Application Tag has a value of FFFFh and the protection information Reference Tag has a value of FFFF_FFFFh.

Inserted protection information consists of the computed CRC-16 in the Guard field, the LBAT field value in the

Application Tag, and the computed reference tag in the Reference Tag field.

Modify a portion of Section 5.14.1.4 (Temperature Threshold) as shown below:

The default value of the over temperature threshold feature for Composite Temperature is the value in the Warning Composite Temperature Threshold (WCTEMP) field in the Identify Controller data if WCTEMP is non-zero; otherwise, it is implementation specific. The default value of the under temperature threshold feature for Composite Temperature is implementation specific. The default value of the over temperature threshold for all implemented temperature sensors is FFFFh. The default value of the under temperature threshold for all implemented temperature sensors is 0h.

If a Get Features command is submitted for this feature, the temperature threshold selected by Command Dword 11 is returned in Dword 0 of the completion queue entry for that command.

Modify a portion of Figure 114 (Temperature Threshold – Command Dword 11) as shown below:

Bit	Description
15:00	**Temperature Threshold (TMPTH):** Indicates the threshold value ~~to apply (in a Set Features) or return (in a Get Features)~~ for the temperature sensor and threshold type specified.

Modify a portion of section 5.14.1.2 (Power Management) as shown below:

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

After a successful completion of a Set Features command for this feature, the controller shall be in the Power State specified. If enabled, autonomous power state transitions continue to occur from the new state.

If a Get Features command is submitted for this Feature, the attributes specified in Figure 111 are returned in Dword 0 of the completion queue entry for that command.

Modify bytes 91:84 in Figure 90 (Identify Controller) as shown below:

87:84	M	**RTD3 Resume Latency (RTD3R):** This field indicates the typical latency in microseconds resuming from Runtime D3 (RTD3). Refer to section 8.4.4 for test conditions. ~~Implementations compliant to revision 1.2 or later of this specification shall report a non-zero value in this field.~~ A value of 0h indicates RTD3 Resume Latency is not reported.
91:88	M	**RTD3 Entry Latency (RTD3E):** This field indicates the typical latency in microseconds to enter Runtime D3 (RTD3). Refer to section 8.4.4 for test conditions. ~~Implementations compliant to revision 1.2 or later of this specification shall report a non-zero value in this field.~~ A value of 0h indicates RTD3 Entry Latency is not reported.

Modify a portion of section 1.5 (Conventions) as shown below:

A 0-based value is a numbering scheme for which the number 0h actually corresponds to a value of 1h and thus produces the pattern of 0h = 1h, 1h = 2h, 2h = 3h, etc. In this numbering scheme, there is not a method for specifying the value of 0h. Values in this specifications are 1-based (i.e., the number 1h corresponds to a value of 1h, 2h=2h, etc.) unless otherwise specified.

Modify a portion of Section 5.10 (Get Log Page command) as shown below:

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

The Get Log Page command uses the PRP Entry 1, PRP Entry 2, and Command Dword 10 fields. All other command specific fields are reserved.

There are mandatory and optional Log Identifiers defined in Figure 76 and Figure 77. If a Get Log Page command is processed that specifies a Log Identifier that is not supported, then the controller should abort the command with status Invalid Field in Command.