LEGAL NOTICE:

This NVM Express Management Interface 1.0a 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 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 - 2018 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 Management Interface Workgroup
c/o NVM Express Administration
3855 SW 153rd Drive
Beaverton, OR 97003
admin@nvmexpress.org

NVM Express™ Technical Errata

Errata ID	**001**
Revision Date	**1/28/2018**
Affected Spec Ver.	**NVM Express™ MI 1.0a**
Corrected Spec Ver.	

Errata Author(s)

Name	Company
Peter Onufryk	Microsemi
Nanci Olson	HPE
Austin Bolen	Dell EMC
Michael Allison	SK hynix

Errata Overview

- Added SMBus clarifications and corrections.
- Corrected errors in VPD NVMe MultiRecord Area and NVMe PCIe Port MultiRecord Area.
- Aligned Product Info Area to IPMI Platform Management FRU Information Storage Definition
- Fixed MIC calculation
- Definitions section is requested to be in alphabetical order.

Revision History

Revision Date	Change Description
5/18/2017	• Initial Errata
6/4/2017	• Split Asset Tag Type/Length/Value into separate fields • Split FRU File ID Type/Length/Value into separate fields • Added Section 2.2 modifications
6/20/2017	• Added a fix for how to reflect the remainder in the Message Integrity Check calculations
6/26/2017	• Made Product Info Area require the format from the Platform Management FRU Information Storage Definition. Added Appendix D to add an example of the Product Info Area.
7/10/2017	• Removed Appendix D. • Added table back into Product Info Area and removed byte offsets • Made explicit requirement on Product Info Area that data conventions do not follow conventions in this document.
7/11/2017	• Adjusted the format of the Product Info Area and included tag, FRU ID, and custom fields.
7/11/2017	• Modified the Product Info Area figure to show what changed using red text + strikethrough. • Added an acronym for the Asset Tag and FRU File ID fields. • Updated errata overview to include MIC fix
7/12/2017	• Clarified the reason to ignore the conventions of this document in section 9.2.2.
7/14/2017	• Added clarification to Product Info Area field descriptions.
10/30/2017	• Editorial changes based on 30-day review.
11/7/2017	• Fixed misspelling in revision history • Removed a change to “a SMBus” • Changed note 2 text color in section 2.2 from red to black since it is existing text • Changed occurrence of “SMBus” to “SMBus/I2C”
1/28/2018	• Updated NVMe administration contact

Description of Specification Changes

Modify Section 1.5.1 as shown below:

Editor's Note:The number of definitions is getting large so they need to be put in alphabetical order.

Modify Section 2.2 as shown below:

If the NVM Subsystem implements an SMBus/I2C interface and associated with that SMBus/I2C interface is a Management Endpoint, then the interface shall support MCTP over SMBus/I2C as specified by the Management Component Transport Protocol (MCTP) SMBus/I2C Transport Binding Specification.

If the NVM Subsystem implements an SMBus/I2C interface, then the NVM Subsystem may optionally support the NVMe Basic Management Command for health and status polling. The NVMe Basic Management Command is defined in Appendix A. It is possible to support both MCTP and the Basic Management Command.

The SMBus/I2C Management Endpoint shall be accessible at a power-up SMBus/I2C address of 3Ah and should be SMBus ARP-capable (as defined in the SMBus 3.0 specification).[1] If the NVM Subsystem is "Discoverable" (as defined in the SMBus 3.0 specification), the device ~~shall~~may issue a "Notify ARP Master" command when the NVM Subsystem is ready to communicate.If the NVM Subsystem implements an SMBus/I2C interface, then VPD information shall be accessible from the Management Endpoint using Sequential Read and Random Read operations as defined by the IPMI Platform Management FRU Information Storage Definition specification.

The VPD shall be accessible using I2C read operations from a FRU Information Device at a power-up SMBus/I2C address of A6h and should be SMBus ARP-capable (as defined in the SMBus 3.0 specification).[2] If the FRU Information Device is "Discoverable" (as defined in the SMBus 3.0 specification), it ~~shall~~may issue a "Notify ARP Master" command when the FRU Information Device is ready to communicate.

If ARP is supported, then the SMBus/I2C Management Endpoint and VPD shall both use the SMBus Address Resolution Protocol Unique Device Identifier (UDID) shown in Figure 7. The only difference between the NVM Subsystem and FRU Information Device UDID is the most significant bit of the Vendor Specific ID. This fact may be used by the MCTP bus owner to associate an SMBus/I2C Management Endpoint with its corresponding VPD.

Clock stretching is allowed by the Management Controller, Management Endpoint, and the VPD. However, implementations are strongly discouraged from using clock stretching so that communications are more predictable with higher throughput.

When a NACK is received, a Management Endpoint shall follow the MCTP specification for a non-bridge endpoint. The Management Endpoint treats a STOP condition due to excessive SMBus NACKs as an implicit Pause Control Primitive. Refer to 4.4.

It is recommended that neither an SMBus Management Endpoint nor a FRU Information Device master the SMBus/I2C or otherwise drive the SMBus/I2C clock or data signals except as required to implement the MCTP over SMBus/I2C Transport Binding Specification.

[1] The address 3Ah appears on SMBus as 0b0011_101x where x represents the SMBus read/write bit.
[2] The address A6h appears on SMBus as 0b1010_011x where x represents the SMBus read/write bit.

Figure 1: NVM Subsystem and FRU Information Device SMBus UDID

Bits	Field	Description
127:120	Device Capabilties	This field describes the device capabilities **Bits** / **Description** 7:6 — **Address Type:** This field describes the type of address contained in the device. Refer to the SMBus transport binding specification. 5:1 — Reserved 0 — **PEC Supported:** All MCTP transactions shall include a Packet Error Code (PEC) byte. This field shall be set to one to indicate support for PEC.
119:112	Version / Revision	This field is used to identify the UDID version and silicon revision. **Bits** / **Description** 7:6 — Reserved 5:3 — **UDID Version.** This field specifies the UDID version and shall be set to 001b 2:0 — **Silicon Revision ID:** This field is used to specify a vendor specific silicon revision level.
111:96	Vendor ID	This field contains the PCI-SIG vendor ID for the Management Endpoint.
95:80	Device ID	This field contains a vendor assigned device ID for the Management Endpoint.
79:64	Interface	This field defines the SMBus version and the Interface Protocols supported. **Bits** / **Description** 15:8 — Reserved 7 — **ZONE.** This field shall be cleared to '0'. 6 — **IPMI.** This field shall be cleared to '0'. 5 — **ASF.** This field shall be set to '1'. Refer to the MCTP transport binding specification. 4 — **OEM.** This field shall be set to '1'. 3:0 — **SMBus Version.** This field shall be set to 4h or 5h which corresponds to SMBus Version 2.0 and 3.0 respectively.
63:48	Subsystem Vendor ID	This field contains the PCI-SIG vendor ID for the Management Endpoint.
47:32	Subsystem Device ID	This field contains a vendor assigned device ID for the Management Endpoint.
31:0	Vendor Specific ID	This field contains a unique 30-bit static NVM storage device ID and is used to distinguish the NVM Subsystem UDID from the FRU Information Device UDID. **Bits** / **Description** 31 — **UDID Type.** This field is used to distinguish the Management Endpoint UDID from the VPD UDID. A '1' in this field indicates the Management Endpoint. A '0' in this field indicates the FRU Information Device. 30 — Reserved. 29:0 — **Unique NVM Storage Device ID:** This field contains a unique vendor assigned ID for the NVM Subsystem. The ID is different in each NVM Subsysteminstance and remains static during the life of the device.

Host platforms expecting to be used with one or more Management Endpoints (e.g., data center platforms and workstations) should isolate SMBus segments to avoid a Management Endpoint conflicting with the address of another SMBus device. An SMBus address conflict may occur when a Management Endpoint is used with platforms that do not isolate SMBus segments (e.g., some client platforms).

Modify Section 3.2.1.1 as shown below:

When sending a message, the Message Integrity Check shall be calculated using the following procedure or a procedure that produces an equivalent result:

1. Initialize the CRC register to FFFFFFFFh. This is equivalent to inverting the lowest 32 bits of the NVMe-

MI Message (Dword 0 in **Error! Reference source not found.**).

2. Append 32 bits of 0's to the end of the Message Data to allow room for the Message Integrity Check (Dword N in **Error! Reference source not found.**). This results in the Message Body shown in **Error! Reference source not found.** with the Message Integrity Check field cleared to 0h.
3. Map the bits in the Message Body 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 Message Body 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 2: Message Integrity Check Example

M(x) =

Message Body (Length = Y bytes)																								
Byte 0								Byte 1								…	Byte Y - 1							
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 1EDC6F41h to produce the 32-bit remainder polynomial R(x).
5. Reflect each byte of R(x) (i.e. bit n of each byte is mapped to bit (~~31~~7 - n) resulting in bit ~~31~~7 to bit 0, bit ~~30~~6 to bit 1, and so on) to produce the polynomial R′(x).
6. Invert R′(x) to produce the polynomial R″(x).
7. Store R″(x) in the Message Integrity Check field of the Message Body.

Upon receipt of an NVMe-MI message, the Message Integrity Check may be validated as follows:

1. Save the received Message Integrity Check.
2. Initialize the CRC register to FFFFFFFFh. This is equivalent to inverting the lowest 32 bits of the NVMe-MI Message (Dword 0 in **Error! Reference source not found.**).
3. Clear the Message Integrity Check field to 0h.
4. Map the bits in the Message Body to the coefficients of the message polynomial M(x) as described in step 3 in the Message Integrity Check calculation procedure above.
5. Divide the polynomial M(x) by the generator polynomial 1EDC6F41h to produce the 32-bit remainder polynomial R(x).
6. Reflect each byte of R(x) (i.e. bit n of each byte is mapped to bit (~~31~~7 - n) resulting in bit ~~31~~7 to bit 0, bit ~~30~~6 to bit 1, and so on) to produce the polynomial R′(x).
7. Invert R′(x) to produce the polynomial R″(x).

Compare R″(x) from step 5 to the Message Integrity Check value saved in step 1. If both values are equal, the Message Integrity Check passes.

Modify Section 9.2.2 as shown below:

Editor's Note: The "section 1.6" in the following paragraph needs to have a reference link added to section 1.6 of this document.

The Product Info Area shall have the same format and conventions as the Product Info Area Format as defined by the IPMI Platform Management FRU Information Storage Definition. Therefore, all fields within the Product Info Area shall not follow the conventions defined in section 1.6. The Product Info Area factory default values shall be set to the values defined in Figure TBD1.

Figure TBD3: Product Info Area Factory Default Values

~~Byte Offset~~	Factory Default	Description
~~00~~	01h	**IPMI Format Version Number (IPMIVER)**: This field indicates the IPMI Format Version.

~~01~~	Impl Spec	**Product Info Area Length (PALEN)**: This field indicates the length of the ~~p~~Product ~~i~~Info ~~a~~Area in multiples of 8 bytes~~ (e.g., 112 bytes/8 = 14 = 0x0Eh)~~.
~~02~~	19h	**Language Code (LCODE):** This field indicates the language used. A value of 19h is used to indicate English.
~~03~~	C8h	**Manufacturer Name Type/Length (MNTL):** This ~~byte~~field indicates the type and length of the Manufacturer Name field.
~~11:04~~	Impl Spec	**Manufacturer Name (MNAME):** This field indicates the Manufacturer name in 8-bit ASCII. Unused bytes should be NULL characters. The Manufacturer name in this field should correspond to that in the PCI Subsystem Vendor ID (SSVID) and IEEE OUI Identifier fields in the Identify Controller Data Structure
~~12~~	D8h	**Product Name Type/Length (PNTL):** This ~~byte~~field indicates the type and length of the Product Name field.
~~36:13~~	Impl Spec	**Product Name (PNAME):** This field indicates the Product name in 8-bit ASCII. Unused bytes should be NULL characters.
~~37~~	E8h	**Product Part/Model Number Type/Length (PPMNNTL):** This ~~byte~~field indicates the type and length of the Product Part/Model Number field.
~~77:38~~	Impl Spec	**Product Part/Model Number (PPMN):** This field indicates the Product Part/Model Number in 8-bit ASCII. Unused bytes should be NULL characters. This field should contain the same value as the Model Number (NM) field in the NVMe Identify Controller Data Structure
~~78~~	C2h	**Product Version Type/Length (PVTL):** This ~~byte~~field indicates the type and length of the Product Part/Model Number field.
~~80:79~~	Impl Spec	**Product Version (PVER):** This field indicates the Product Version in 8-bit ASCII. Unused bytes should be NULL characters.
~~81~~	D4h	**Product Serial Number Type/Length (PSNTL):** This ~~byte~~field indicates the type and length of the Product Serial Number field.
~~101:82~~	Impl Spec	**Product Serial Number (PSN):** This field indicates the Product Serial Number in 8-bit ASCII. Unused bytes should be NULL characters. This field should contain the same value as the Serial Number (SN) field in the NVMe Identify Controller Data Structure.
~~102~~	~~0h~~Impl Spec	**Asset Tag Type/Length (ATTL):** This ~~byte~~field indicates the type and length of the Asset Tag field. A value of 00h may be used to indicate an Asset Tag is not present.
	Impl Spec	**Asset Tag (AT):** This field indicates the asset tag.
~~103~~	~~0h~~Impl Spec	**FRU File ID Type/Length (ATTL):** This ~~byte~~field indicates the type and length of the FRU File ID field. A value of 00h may be used to indicate a FRU File ID is not present.
	Impl Spec	**FRU File ID (FFI):** This field provides manufacturing aid for verifying the file that was used during manufacture or field update to load the FRU information.
	Impl Spec	**Custom Product Info Area (CPIA):** This optional field allows for the addition of custom Product Info Area fields that shall be proceeded with a Type/Length field.
~~104~~	C1h	**End of Record (EOR)**: A value of C1h in this ~~byte~~field indicates the end of record
~~110:105~~	0h	~~Reserved~~Zero or more bytes of value 0h that are used to pad the size of the Product Info Area to a multiple of 8 bytes.
~~111~~	Impl Spec	**Product Info Area (PICHK):** Checksum computed over all bytes~~ 0 through 110~~ in the Product Info Area excluding this field. The checksum is computed by adding the 8-bit value of the bytes modulo 256 and then taking the 2's complement of this sum. When the checksum and the sum of the bytes module 256 are added, the result should be 0h.

Modify Section 9.2.3 as shown below:

9.2.3 NVMe MultiRecord Area

Byte Offset	Factory Default	Description
00	0Bh	**NVMe Record Type ID**
01	~~2h~~ Impl Spec	Bit 7 – end of list; record format version = 2h The value of this field is 82h if this is the last record in the list and 2h if it is not the last record in the list.
02	~~28h~~3Bh	**Record Length (RLEN)**: This field indicates the number of bytes of data in the record. ~~length of the MultiRecord Area in bytes.~~
03	Impl Spec	**Record Checksum**: This field is used to give the record data a zero checksum (i.e., the modulo 256 sum of the record data bytes from byte offset 05 through the end of this record plus this checksum byte equals zero)

Modify Section 9.2.4 as shown below:

9.2.4 NVMe PCIe Port MultiRecord Area

Byte Offset	Factory Default	Description
00	0Ch	**NVMe PCIe Port Record Type ID**
01	~~2h~~ Impl Spec	Bit 7 – end of list; record format version = 2h The value of this field is 82h if this is the last record in the list and 2h if it is not the last record in the list.
02	~~28h~~0Bh	**Record Length (RLEN)**: This field indicates the number of bytes of data in the record. ~~length of the MultiRecord Area in bytes.~~
03	Impl Spec	**Record Checksum**: This field is used to give the record data a zero checksum (i.e., the modulo 256 sum of the record data bytes from byte offset 05 through the end of this record plus this checksum byte equals zero)