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	**001**
Revision Date	**10/24/2016**
Affected Spec Ver.	**NVMe over Fabrics 1.0**
Corrected Spec Ver.	

Errata Author(s)

Name	Company
Judy Brock	Samsung
David Black	EMC
James Smart	Broadcom
Fred Knight	NetApp

Errata Overview

The erratum includes clarifications for NVMe over Fabrics 1.0, including:

- Clarifying that the Connect command creates the queue that it is “sent” on.
- Adding a theory of operation section that mirrors the section 4.1 for queues and doorbells in the Base specification.
- Various editorial updates and clarifications.

Revision History

Revision Date	Change Description
5/31/2016	Initial draft, including Connect updates proposed by Judy.
6/9/2016	Included theory of operation for queues in Fabrics proposed by Judy, Ken, and David.
6/16/2016	Updates based on feedback in 6/9 meeting and email from Fred Knight.
6/24/2016	Simplify Submission Queue Entry lifetime text
7/5/2016	Keep Alive clarifications for enabled controllers
7/14/2016	Add RDMA discovery info, move 1.2.1 changes to separate ECN doc
7/19/2016	Removed items suggested for ECN 002 based on 7/14 meeting.
7/20/2016	Minor wording edit.
8/8/2016	Red-line accept of 7/20 version.
10/24/2016	Errata ratified.

Description of Specification Changes

Modify a portion of section 1.5.7 as shown below:

NVMe over Fabrics uses the Connect command to create controller Admin or I/O Queues. The creation of an Admin Queue establishes an association between a host and the corresponding controller. NVMe over Fabrics does not support the Admin Submission Queue Base Address (ASQ), Admin Completion Queue Base Address (ACQ), and Admin Queue Attributes (AQA) properties as all information necessary to establish an Admin Queue is contained in the Connect command. NVMe over Fabrics does not support the Admin commands associated with I/O Queue creation and deletion (Create I/O Completion Queue, Create I/O Submission Queue, Delete I/O Completion Queue, Delete I/O Submission Queue) defined in the NVMe Base specification.

An NVMe Transport connection is established between a host and an NVM subsystem prior to the transfer of any capsules or data. The mechanism used to establish an NVMe Transport connection is NVMe Transport specific and defined by the corresponding NVMe Transport binding specification. The NVMe Transport may require a separate NVMe Transport connection for each Admin or I/O Queue or may utilize the same NVMe Transport connection for all Admin and I/O Queues associated with a particular controller. An NVMe Transport may also require that NVMe layer information be passed between the host and controller in the process of establishing an NVMe Transport connection (e.g., exchange queue size to appropriately size send and receive buffers).

The Connect command specifies the Queue ID and type (Admin or I/O), the size of the Submission and Completion Queues, queue attributes, Host NQN, NVM Subsystem NQN, and Host Identifier. The Connect command may specify a particular controller if the NVM subsystem supports a static controller model. The Connect response indicates whether the connection was successfully established as well as whether NVMe in-band authentication is required.

The Connect command is submitted to the same Admin Queue or I/O Queue that it creates. The underlying NVMe Transport connection that is used for that queue is created first and the Connect command and response capsules are sent over that NVMe Transport connection. The Connect command may only be sent once to a queue.

When a Connect command successfully completes, the corresponding Submission and Completion Queues are created. If NVMe in-band authentication is required as indicated in the Connect response, then NVMe in-band authentication shall be performed before the queues may be used to perform other Fabrics, Admin, or I/O commands. Once a Connect command for an Admin Queue has completed successfully (and NVMe in-band

authentication if required), only Fabrics commands may be submitted until the controller is ready (CSTS.RDY = 1). Both Fabrics and Admin commands may be submitted to the Admin Queue while the controller is ready. Once a Connect command for an I/O Queue has completed successfully (and NVMe in-band authentication if required), I/O commands may be submitted to the queue.

The Connect response contains the controller ID allocated to the host. All subsequent Connect commands that create an I/O Queue with that controller shall be from the same host, utilize the same NVMe Transport, and have the same Host Identifier, Host NQN, and NVM Subsystem NQN; if any of these conditions do not hold then the Connect command fails.

Modify Figure 14 as shown below:

Figure 14: Fabric Command Types

Command Type by Field			Combined Command Type[2]	O/M[1]	I/O Queue[3]	Command
(07)	(06:02)	(01:00)				
Generic Command	Function	Data Transfer[4]				
0b	000 00b	00b	00h	M	No	Property Set
0b	000 00b	01b	01h	M	Yes	Connect[5]
0b	000 01b	00b	04h	M	No	Property Get
0b	000 01b	01b	05h	O	Yes	Authentication Send
0b	000 01b	10b	06h	O	Yes	Authentication Receive
	Vendor Specific					
1b	na	na	C0h – FFh	O		Vendor specific

NOTES:
1. O/M definition: O = Optional, M = Mandatory.
2. Opcodes not listed are reserved.
3. All Fabrics commands may be submitted on the Admin Queue. The I/O Queue supports Fabrics commands as specified in this column.
4. 00b = no data transfer; 01b = host to controller; 10b = controller to host; 11b = reserved
5. The Connect command is submitted and completed on the same queue that it creates. Refer to section 1.5.7.

Modify a portion of section 3.3 as shown below:

The host shall establish an association with a controller and enable the controller before establishing a connection with an I/O Queue of the controller. If the host sends a Connect command specifying a Queue ID for an I/O Queue before the controller has been enabled, then a status value of Connect Invalid Parameters is returned. If the host sends a Connect command specifying a Queue ID for an Admin or I/O Queue which has already been created, then a status value of Command Sequence Error is returned.

Modify a portion of section 4.3 as shown below:

When a Connect command successfully completes, the corresponding Admin Submission and Completion Queue or I/O Submission and Completion Queues are created. If the host sends a Connect command specifying the Queue ID of a queue which already exists, then a status value of Command Sequence Error is returned.

The Authentication Requirements (AUTHREQ) field in the Connect response indicates if NVMe in-band authentication is required. If AUTHREQ is cleared to zero, the created queue is ready for use after the Connect command completes successfully. If AUTHREQ is set to a non-zero value, the created queue is ready for use after NVMe in-band authentication has been performed successfully using the Authentication Send and Authentication Receive Fabrics commands.

If a controller requires or is undergoing NVMe in-band authentication, a controller shall abort all commands other than authentication commands with a status of Authentication Required. After the NVMe in-band authentication has been performed successfully, a controller shall abort all authentication commands with a status of Command Sequence Error.

When an Admin Queue is ready for use, the associated controller is disabled (i.e., CC.EN is initialized to '0'). A disabled controller shall abort all commands other than the Property Get and Property Set commands on the Admin Queue with a status of Command Sequence Error. After the controller is enabled, it should accept all supported Admin commands in addition to the Fabrics commands Property Get and Property Set.

A created I/O queue shall abort all commands with a status of Command Sequence Error if the associated controller is disabled.

Add section 1.4.14 as shown below:

1.4.14 command submission

A command is submitted when a host adds a capsule to a Submission Queue. The host increments the Tail entry pointer associated with that Submission Queue as part of submitting a command.

Replace section 2.4 as shown below:

~~**2.4 Queue Flow Control Mechanism**~~
2.4 Submission Queue and Completion Queue Definition

NVMe over Fabrics Submission Queues and Completion Queues are message-based (refer to Figure 1) in contrast to NVMe over PCIe memory-based queues (refer to section 4.1 in NVM Express 1.2.1), Doorbells are not used by NVMe over Fabrics. For the remainder of this section, the terms Submission Queue, Completion Queue and queue refer to NVMe over Fabrics queues unless explicitly stated otherwise.

For NVMe over Fabrics, a queue is a unidirectional communication channel that is used to send capsules between a host and a controller. A host uses Submission Queues to send command capsules (refer to section 2.1) to a controller. A controller uses Completion Queues to send response capsules (refer to section 2.2) to a host. Submission and Completion Queues are created in pairs using the Connect command (refer to section 1.5.7).

Each Submission Queue has a Head entry pointer and a Tail entry pointer that are used to manage the queue and determine the number of outstanding capsules. The Head and Tail entry pointers are initialized to zero when a queue is created. All arithmetic operations and comparisons on entry pointers are performed modulo the queue size with queue wrap conditions taken into account. The host increments the Tail entry pointer when it adds a capsule to a queue. The controller increments the Head entry pointer when it removes a capsule from the queue.

The Submission Queue is empty when the Head entry pointer equals the Tail entry pointer. A capsule consumer may continue to remove capsules from the queue as long as the empty queue condition is not met.

The Submission Queue is full when the Head entry pointer equals one more than the Tail entry pointer (i.e., incrementing the Tail entry pointer has caused it to wrap around to just behind the Head entry pointer). A full Submission Queue contains one less capsule than the queue size. A host may continue to add capsules to a Submission Queue as long as the queue is not full.

~~Submission Queue flow control is facilitated by the use of the SQHD field in the Completion Queue Entry. The controller uses the SQHD field to communicate the availability of Submission Queue slots to the host.~~

Completion Queues do not use Head entry pointers or Tail entry pointers (refer to section 2.4.2).

The definition for the queue attributes of Queue Size, Queue Identifier and Queue Priority are defined in sections 4.1.3, 4.1.4 and 4.1.5 of NVM Express 1.2.1.

NVMe Transports are not required to provide any additional end-to-end flow control. Specific NVMe Transports may require low level flow control for congestion avoidance and reliability; any such additional NVMe Transport flow control ~~which~~ is outside the scope of this specification.

Flow control differs for Submission Queues and Completion Queues (refer to sections 2.4.1 and 2.4.2).

2.4.1 Submission Queue Flow Control

An NVMe over Fabrics Submission Queue Tail entry pointer is local to the host and is not communicated to the controller. The NVMe Transport is responsible for promptly delivering command capsules to the controller and notifying the controller of capsule arrival in a transport-specific fashion.

The NVMe over Fabrics Submission Queue Head entry pointer is maintained by the controller and is communicated to the host in the SQHD field of Completion Queue Entries. The host uses the received SQHD values for Submission Queue management (e.g., to determine whether the Submission Queue is full).

Altering a command capsule between host submission to the Submission Queue and transport delivery of that capsule to the controller results in undefined behavior.

If the controller detects that the host has submitted ~~the host submits~~ a command capsule to a full ~~when no~~ Submission Queue, ~~slots are available~~ then the controller shall stop processing commands and set the Controller Fatal Status (CSTS.CFS) bit to '1' (refer to section 9.5 in the NVMe Base specification).

2.4.2 Completion Queue Flow Control Considerations

Completion Queue flow control is not used in NVMe over Fabrics. NVMe over Fabrics Completion Queues do not use either Head entry pointers or Tail entry pointers.

The host should size each Completion Queue to support the maximum number of commands that it ~~has~~ could have outstanding at one time for a particular Submission Queue ~~queue~~. The Completion Queue size may be larger than the size of the corresponding Submission Queue to accommodate responses for commands that are being processed by the controller in addition to responses for commands are still in the Submission Queue.

If the size of a Completion Queue is too small for the number of outstanding commands and the controller submits a response capsule to a full Completion Queue, then the results are undefined.

The Maximum Outstanding Commands (MAXCMD) value in the Identify Controller data structure indicates the maximum number of commands that the controller processes at one time for a particular queue ~~to achieve the best performance~~. The host may use this value to size Completion Queues and optimize the number of commands submitted at one time per queue to achieve the best performance.

Altering a response capsule between controller submission to the Completion Queue and transport delivery of that capsule to the host results in undefined behavior.

Modify the definition of "fabric" in section 1.4.7 as shown below:

A network topology in which nodes pass data to each other ~~through interconnecting switches~~.

Modify a portion of section 1.5.5 as shown below:

Properties are the NVMe over Fabrics analog of memory mapped NVMe controller registers defined for NVMe over PCIe. Properties are used to configure a subset of ~~low level~~ controller attributes and obtain a subset of ~~low level~~ status.

Modify a portion of section 1.5.7 as shown below:

When a Connect command successfully completes, the corresponding Submission and Completion Queues are created. If NVMe in-band authentication is required as indicated in the Connect response, then NVMe in-band authentication shall be performed before the queues may be used to perform other Fabrics, Admin, or I/O commands. Once a Connect command for an Admin Queue has completed successfully (and NVMe in-band authentication if required has succeeded), only Fabrics commands may be submitted until the controller is ready (CSTS.RDY = 1). Both Fabrics and Admin commands may be submitted to the Admin Queue while the controller is ready. A Connect command for an I/O Queue may only be submitted after the controller is ready. Once a Connect command for an I/O Queue has completed successfully (and NVMe in-band authentication if required has succeeded), I/O commands may be submitted to the queue.

Modify a portion of section 2.3.1 as shown below:

- the host shall not place more SGL Data Block or Keyed SGL Data Block descriptors ~~(Keyed) Data Block SGLs~~ within a capsule than the maximum indicated in the Identify Controller data structure.

Modify a portion of Figure 15 as shown below:

39:24	**SGL Descriptor 1 (SGL1):** This field contains the first SGL descriptor for the command. If the SGL descriptor is an SGL Data Block or Keyed SGL ~~a (Keyed)~~ Data Block descriptor, then it describes the entire data transfer. If more than one SGL descriptor is needed to describe the data transfer, then the first SGL descriptor is a Segment or Last Segment descriptor. Refer to section 4.4 of the NVMe Base specification for the definition of SGL descriptors.

Modify a portion of Figure 17 as shown below:

39:24	**SGL Descriptor 1 (SGL1):** This field contains the first SGL descriptor for the command. If the SGL descriptor is an SGL Data Block or Keyed SGL ~~a (Keyed)~~ Data Block descriptor, then it describes the entire data transfer. If more than one SGL descriptor is needed to describe the data transfer, then the first SGL descriptor is a Segment or Last Segment descriptor. Refer to section 4.4 of the NVMe Base specification for the definition of SGL descriptors.

Modify a portion of Figure 19 as shown below:

39:24	**SGL Descriptor 1 (SGL1):** This field contains an SGL Data Block or Keyed ~~a (Keyed)~~ SGL Data Block descriptor that describes the entire data transfer. Refer to section 4.4 of the NVMe Base specification for the definition of SGL descriptors.

Modify a portion of Figure 28 as shown below:

1803	M	**Maximum SGL Data Block Descriptors (MSDBD):** This field indicates the maximum number of ~~(Keyed)~~ SGL Data Block or Keyed SGL Data Block descriptors that a host is allowed to place in a capsule. A value of 0h indicates no limit.

Modify a portion of section 7.3.2 as shown below:

Admin command data is transferred using host-resident data buffers specified in Keyed SGL Data Block descriptor entries. I/O command data is transferred using host-resident data buffers specified in Keyed SGL Data Block descriptor entries or within the capsule. The RDMA Transport supports the SGL Data Block, SGL Last Segment, and Keyed SGL Data Block descriptors only. The RDMA Transport does not support SGLs in host memory; all SGLs shall be contained in the command capsule. Fabrics and Admin commands have one (Keyed) SGL Data Block or Keyed SGL Data Block descriptor (i.e., there are no SGL descriptors following the Submission Queue Entry). I/O commands may have more than one SGL descriptor.

Modify a portion of Figure 28 as shown below:

1802	M	**Fabrics Controller Attributes (FCATT ~~CTRATTR~~):** This field indicates attributes of the controller that are specific to NVMe over Fabrics. Bits 7:1 are reserved. Bit 0 if cleared to '0' then the NVM subsystem uses a dynamic controller model. Bit 0 if set to '1' then the NVM subsystem uses a static controller model.

Modify section 4.3 as shown below:

When an Admin Queue is ~~ready for use~~ first created, the associated controller is disabled (i.e., CC.EN is initialized to '0'). A disabled controller shall abort all commands other than ~~the Property Get and Property Set~~ Fabrics commands on the Admin Queue with a status of Command Sequence Error. After the controller is enabled, it ~~should~~ shall accept all supported Admin commands in addition to ~~the~~ Fabrics commands ~~Property Get and Property Set~~.

Modify a portion of section 7.1.1 as shown below:

An NVMe Transport may transmit an SQHD value in every response capsule. If an NVMe Transport does not transmit an SQHD value in every response capsule, then an SQHD value should be transmitted periodically (e.g., in at least one of every n response capsules on a CQ, where n is 10% of the size of the associated SQ) or more often. An SQHD value should always be transmitted if 90% or more of the slots in the associated SQ are occupied at the subsystem~~, or if the associated SQ is empty at the subsystem~~.

Modify a portion of Figure 28 as shown below:

767:512	**Transport Address (TRADDR):** Specifies the address of the NVM subsystem that may be used for a Connect command as an ASCII string. The Address Family field describes the reference for parsing this field. Refer to section 1.5 of the NVMe Base specification for ASCII string requirements. For the definition of this field, refer to the appropriate NVMe Transport binding specification.

Modify a portion of Figure 34 as shown below:

31:10	Reserved
63:32	**Transport Service Identifier (TRSVCID):** Specifies the NVMe Transport service identifier as an ASCII string. The NVMe Transport service identifier is specified by the associated NVMe Transport binding specification. ~~Refer to http://nvmexpress.org/specifications for a registry that lists the valid values of this identifier and the associated NVMe Transport binding specifications.~~
255:64	Reserved