FC-NVMe rev 1.14 (T11/16-020vB) Internal Review Comment Summary

(April 19, 2017)

Table 1 Comment Breakdown

Company	Technical	Editorial	Total
Dell	~138	~83	223
IBM	~139	~116	268
NetApp	?	?	116
Oracle	0	2	2
QLogic	71	62	133
Broadcom	24	149	173
Viavi	4	66	70
Brocade	2	8	10
Total			**~997**

Table 2 Comment Database Metrics

Date	:A: Accept	:AinP: Accept in Principle	:AI: Action Item	:C: Complete	:O: Open	:R: Reject	:T: Technical	:E: Editorial	PDF total
3/21	69	37	2	99	2	5	423	560	995
3/22	82	44	6	99	33	6	423	560	997
4/19	116	154	1	270	16	14	424	560	997

Copies of this document may be purchased from:
Global Engineering, 15 Inverness Way East,
Englewood, CO 80112-5704
Phone: (800) 854-7179 or (303) 792-2181 Fax: (303) 792-2192

FIBRE CHANNEL

NVME

(FC-NVMe)

REV 1.14

INCITS working draft proposed
American National Standard
for Information Technology

December 7, 2016

Secretariat: Information Technology Industry Council

NOTE:
This is a working draft American National Standard of Accredited Standards Committee INCITS. As such this is not a completed standard. Representatives of the T11 Technical Committee may modify this document as a result of comments received anytime, or during a future public review and its eventual approval as a Standard. Use of the information contained herein is at your own risk.

POINTS OF CONTACT:

Steven Wilson (T11 Chair)
Brocade Communcations, Inc.
130 Holger Way
San Jose, CA 95134
Voice: 408-333-8128
swilson@brocade.com

Claudio Desanti (T11 Vice Chair)
Cisco Systems, Inc.
170 W. Tasman Dr.
San Jose, CA 95134
Voice: 408-853-9172
cds@cisco.com

Craig W. Carlson (T11.3 Chair)
QLogic Corporation
46)01 Dean Lakes Blvd
Shakopee, MN 55379
Voice: 952-687-2431
craig.carlson@qlogic.com

Craig Carlson (FC-NVMe Chair)
QLogic Corporation
46)01 Dean Lakes Blvd
Shakopee, MN 55379
Voice: 952-687-2431
craig.carlson@qlogic.com

David Peterson (FC-NVMe Editor)
Brocade Communications, Inc.
130 Holger Way, CA
San Jose, CA 95134
Voice: 763-248-9374
david.peterson@brocade.com

Change History

Rev 1.14
16-483v1 - WWN uniqueness
Modified clause 12 - Timers for operation and recovery
16-211v6 - clause 11 - Link error detection and recovery procedures
16-518v0 - NVMe_RJT reason and explanation codes
16-479v3 - Discovery and IU exchange

Rev 1.13
Removed Sequence level error detection and recovery.
Incorporated 16-473v2 Clause 4 Associations and Connections
Incorporated 16-326v5 diagrams 1-3
16-476v3 - Draft standard updates, excluding clause 12 Timers

Rev 1.12
16-466v1 - Clause 8 - FC-4 Link Services updates
16-465v1 - Clause 9 - Information Unit updates
16-467v0 - Clause 10 - NVMe over Fabrics updates

Rev 1.11
16-418v5 - Clause 4 - General updates
16-461v0 - Read DATA IU loss detection

Rev 1.10
16-337v3 - NVMe over Fabrics updates
16-336v5 - FC-4 Link Service updates
16-390v3 - Link Service updates
16-450v2 - Link Service updates
16-432v0 - Added two reserved words to end of NVMe_CMND IU

Rev 1.09
16-388v0 - FC-NVMe: Information Unit updates

Rev 1.08
16-154v2 - FC-NVMe: Data Transfer Rules

Rev 1.07
16-200v1 - FC-NVMe: Discovery - Who are you again?
16-230v0 - FC-NVMe: NVMe over Fabrics updates
16-247v0 - FC-NVME IU payload Endianness
16-108v4 - FC-NVMe: Structure and concepts

Rev 1.06
16-156v2 - FC-NVMe: A New Order Detailed Text
16-214v1 - FC-NVMe: FC-4 Name Server registration and objects
16-188v1 - FC-NVMe: Link Services updates

Rev 1.05
16-199v2 - FC-NVMe: FC-4 Link Service updates

Rev 1.04
16-019v2 - FC-NVMe: FC-4 Link Service updates

Rev 1.03

BSR INCITS 540-201x

American National Standard
for Information Technology

Fibre Channel - NVMe (FC-NVMe)

Secretariat

Information Technology Industry Council

Approved (not yet approved)

American National Standards Institute, Inc.

Abstract

This standard describes the frame format and protocol definitions required to transfer commands and data between a NVM Express host and NVM Express subsystem using the Fibre Channel family of standards.

American National Standard

Approval of an American National Standard requires review by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer.

Consensus is established when, in the judgement of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that a concerted effort be made towards their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and under no circumstance gives an interpretation of any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

PATENT STATEMENT

The developers of this standard have requested that holders of patents that may be required for the implementation of the standard disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard. As of the date of publication of this standard, following calls for the identification of patents that may be required for the implementation of the standard, notice of one or more such claims has been received. By publication of this standard, no position is taken with respect to the validity of this claim or of any rights in connection therewith. The known patent holder(s) has (have), however, filed a statement of willingness to grant a license under these rights on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such a license. Details may be obtained from the publisher. No further patent search is conducted by the developer or publisher in respect to any standard it processes. No representation is made or implied that this is the only license that may be required to avoid infringement in the use of this standard.

Published by

American National Standards Institute
11 West 42nd Street, New York, NY 10036

Printed in the United States of America

Foreword (This Foreword is not part of American National Standard INCITS 540-201x.)

This standard defines a Fibre Channel mapping layer (FC-4) that uses the services defined by INCITS Project 545-D, Fibre Channel Framing and Signaling Interface - 5 (FC-FS-5) to transmit command, data, and status information between an NVMe host and an NVM subsystem. The use of the standard enables the transmission of standard NVMe command formats, the transmission of standard NVMe data and control, and the receipt of NVMe status across the Fibre Channel using standard Fibre Channel frame and Sequence formats. The NVMe protocol operates with Fibre Channel Class 3 Service, and operates across Fibre Channel fabrics.This standard was developed by Task Group T11.3 of Accredited Standards Organization INCITS during 2014-201x. The standards approval process started in 201x. This document includes annexes that are informative and are not considered part of the standard.

Requests for interpretation, suggestions for improvements or addenda, or defect reports are welcome. They should be sent to the INCITS Secretariat, Information Technology Industry Council, 1101 K Street, NW Suite 610, Washington, DC 20005.

This standard was processed and approved for submittal to ANSI by the International Committee for Information Technology Standards (INCITS). Committee approval of the standard does not necessarily imply that all committee members voted for approval.

At the time it approved this standard, INCITS had the following members:

(to be filled in by INCITS)

Technical Committee T11 on Fibre Channel Interfaces, which reviewed this standard, had the following members:

[to be filled in prior to publication]

Task Group T11.3 on Interconnection Schemes, which developed and reviewed this standard, had the following members:

[to be filled in prior to publication]

Introduction

FC-NVMe defines a mapping protocol for applying the NVM Express interface to Fibre Channel. This standard defines how Fibre Channel services and specified Information Units (IUs) are used to perform the services defined by the NVM Express interface specification.

Contents **Page**

Figure **Page**

Table **Page**

American National Standard
for Information Technology —

Fibre Channel —
NVMe (FC-NVMe)

1 Scope

This standard defines a mapping protocol for applying the NVM Express over Fabrics interface to Fibre Channel. This standard defines how the Fibre Channel services and the defined Information Units (IUs) are used to perform the services defined by the NVM Express over Fabrics interface specification.

2 Normative References

2.1 Overview

The following standards contain provisions that, through reference in the text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

For electronic copies of ANSI and INCITS standards, visit ANSI's Electronic Standards Store (ESS) at http://www.ansi.org. For printed versions of most standards listed here, contact Global Engineering Documents, 15 Inverness Way East, Englewood, CO; 80112-5704, (800) 854-7179.

Orders for ISO Standards and ISO publications should normally be addressed to the ISO member in your country. If that is impractical, ISO Standards and ISO publications may be ordered from ISO Central Secretariat (ISO/CS):

Phone +41 22 749 01 11
Fax +41 22 749 09 47
E-mail sales@iso.org
Post ISO, 1, rue de Varembé, CH-1211
Geneva 20, Switzerland

In order to avoid delivery errors, it is important that you accurately quote the standard's reference number given in the ISO catalogue. For standards published in several parts, you should specify the number(s) of the required part(s). If not, all parts of the standard will be provided.

Copies of the following documents may be obtained from ANSI, an ISO member organization:

a) approved ANSI standards;

b) approved and draft international and regional standards (ISO and IEC); and

c) approved foreign standards (JIS and DIN).

For further information, contact the ANSI Customer Service Department:

Phone +1 212-642-4900
Fax: +1 212-302-1286
Web: http://www.ansi.org
E-mail: ansionline@ansi.org

or the InterNational Committee for Information Technology Standards (INCITS):

Phone 202-626-5738
Web: http://www.incits.org
E-mail: incits@itic.org

Additional availability contact information is provided below as needed.

2.2 Approved references

2.3 References under development

At the time of publication, the following referenced standards were still under development. For information on the current status of the documents, or regarding availability, contact the relevant standards body or other organization as indicated.

For electronic copies of references under development by INCITS T11, see www.t11.org.

T11/Project 545-D, *Fibre Channel - Framing and Signaling - 5 (FC-FS-5)*

T11/Project 547-D, *Fibre Channel - Switch Fabric - 7 (FC-SW-7)*

T11/Project 2237-D, *Fibre Channel - Link Services - 3 (FC-LS-3)*

T11/Project 548-D, *Fibre Channel - Generic Services - 8 (FC-GS-8)*

T10/Project 546-D, *SCSI Architecture Mode - 6 (SAM-6)*

2.4 NVM Express references

Copies of the following approved NVM Express standards may be obtained through the NVM Express organization at http://nvmexpress.org.

NVM Express revision 1.2.1 - June 5, 2016

NVMe over Fabrics revision 1.0 - June 5, 2016

3 Definitions and conventions

3.1 Overview

For FC-NVMe, the following abbreviations, acronyms, definitions, conventions, symbols, and keywords apply.

3.2 Abbreviations and acronyms

Abbreviations and acronyms applicable to this standard are listed. Definitions of several of these items are included in 3.3.

BLS Basic Link Service
CQE Completion Queue Entry
ELS Extended Link Service
FC-FS-5 Fibre Channel - Framing and Signaling - 5
FC-GS-8 Fibre Channel - Generic Services - 8
FC-LS-3 Fibre Channel - Link Services - 3
FC-SP-2 Fibre Channel - Security Protocols - 2
FC-SW-7 Fibre Channel - Switched Fabric - 7
FLOGI Fabric Login
IU Information Unit
LS_ACC Link Service Accept reply frame
LS_RJT Link Service Reject reply frame
lsb least significant bit
LSB least significant byte
msb most significant bit
MSB most significant byte
NVMe NVM Express
NPBSY N_Port Busy
PLOGI N_Port Login
PRLI Process Login
SGL Scatter Gather List
SQE Submission Queue Entry

3.3 Common definitions

3.3.1 capsule

NVMe unit of information exchange used in NVMe over Fabrics

Note 1 to entry: See NVMe over Fabrics revision 1.0.

3.3.2 Data Series

set of NVMe_DATA IUs that make up the total data transfer for a particular command

3.3.3 FC_Port

port capable of transmitting and receiving Fibre Channel frames

Note 1 to entry: See FC-FS-5.

3.3.4 FLOGI

Fabric Login ELS

Note 1 to entry: See FC-LS-3.

3.3.5 LBA data
data read from or written to NVMe storage device

3.3.6 LS_ACC
Link Service Accept

Note 1 to entry: See FC-LS-3.

3.3.7 LS_RJT
Link Service Reject

Note 1 to entry: See FC-LS-3.

3.3.8 metadata
contextual information about particular LBA data

Note 1 to entry: See NVM Express revision 1.2.1.

3.3.9 Name_Identifier
64-bit identifier, with a 60-bit value preceded with a 4-bit Network_Address_Authority Identifier, used to identify entities in Fibre Channel (e.g., N_Port, node, F_Port, or Fabric)

Note 1 to entry: See FC-FS-5.

3.3.10 Node_Name
Name_Identifier (see 3.3.9) associated with a node

Note 1 to entry: See FC-FS-5.

3.3.11 NVMe connection
relationship between an NVMe host and a particular NVM subsystem, controller, and Queue

3.3.12 NVMe controller
entity that implements the NVMe function

Note 1 to entry: See NVM Express revision 1.2.1.

3.3.13 NVMe_Port
Nx_Port that supports the Fibre Channel NVM Express over Fabrics protocol

3.3.14 N_Port
device port that generates or terminates FC-4 traffic

3.3.15 N_Port_ID
topology unique address identifier of an N_Port

Note 1 to entry: See FC-FS-5.

3.3.16 N_Port_Name
Name_Identifier (see 3.3.9) that identifies an N_Port (see 3.3.14)

3.3.17 PLOGI
N_Port Login

Note 1 to entry: See FC-LS-3.

3.3.18 BSY

N_Port Busy

Note 1 to entry: See FC-FS-5.

3.3.19 SGL data

data pointed to by an SGL

3.4 Editorial conventions

In FC-NVMe, a number of conditions, mechanisms, sequences, parameters, events, states, or similar terms are printed with the first letter of each word in uppercase and the rest lowercase (e.g., Exchange, Sequence). Any lowercase uses of these words have the normal technical English meanings.

Lists sequenced by letters (e.g., a-red, b-blue, c-green) show no ordering relationship between the listed items. Numbered lists (e.g., 1-red, 2-blue, 3-green) show an ordering relationship between the listed items.

In case of any conflict between figure, table, and text, the text, then tables, and finally figures take precedence. Exceptions to this convention are indicated in the appropriate clauses.

In all of the figures, tables, and text of this document, the most significant bit of a binary quantity is shown on the left side. Exceptions to this convention are indicated in the appropriate clauses.

Data structures in this standard are displayed in Fibre Channel format (i.e., “big-endian”), while specifications originating in NVMe over Fabrics display data structures in Ethernet format (i.e., “little-endian”).

If the value of the bit or field is not relevant, then x or xx appears in place of a specific value. If a field or a control bit in a frame is specified as not meaningful, then the entity that receives the frame shall not check that field or control bit.

Numbers that are not immediately followed by lower-case b or h are decimal values.

Numbers immediately followed by lower-case b (xxb) are binary values.

Numbers or upper case letters immediately followed by lower-case h (xxh) are hexadecimal values.

In figures, dashed components or bracketed components are optional.

3.5 Symbols

Unless indicated otherwise, the following symbol has the listed meaning.

!= not equal

3.6 Keywords

3.6.1 ignored: A keyword used to describe an unused bit, byte, word, field or code value. The contents or value of an ignored bit, byte, word, field or code value shall not be examined by the receiving device and may be set to any value by the transmitting device.

3.6.2 invalid: A keyword used to describe an illegal or unsupported bit, byte, word, field or code value. Receipt of an invalid bit, byte, word, field or code value shall be reported as an error.

3.6.3 mandatory: A keyword indicating an item that is required to be implemented as defined in this standard.

3.6.4 may: A keyword that indicates flexibility of choice with no implied preference (equivalent to “may or may not”).

3.6.5 may not: A keyword that indicates flexibility of choice with no implied preference (equivalent to “may or may not”).

3.6.6 optional: A keyword that describes features that are not required to be implemented by this standard. However, if any optional feature defined by this standards is implemented, then it shall be implemented as defined in this standard.

3.6.7 reserved: A keyword referring to bits, bytes, words, fields and code values that are set aside for future standardization. A reserved bit, byte, word or field shall be set to zero, or in accordance with a future extension to this standard. Recipients should not check reserved bits, bytes, words or fields for zero values. Receipt of reserved code values in defined fields shall be reported as an error.

3.6.8 shall: A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard.

3.6.9 should: A keyword indicating flexibility of choice with a strongly preferred alternative; equivalent to the phrase “it is strongly recommended”.

3.6.10 x or **xx:** The value of the bit or field is not relevant.

4 General

4.1 Structure and concepts

Fibre Channel (FC) is logically a point-to-point serial data channel. The architecture has been designed so that it may be implemented with high performance hardware that requires little real-time software management. The Fibre Channel Physical layer (FC-2 layer) described by FC-FS-5 performs those functions required to transfer data from one N_Port to another. In this standard, N_Ports capable of supporting NVMe over FC transactions are collectively referred to as NVMe_Ports. The FC-2 layer is a delivery service with information grouping and defined classes of service.

A switching fabric allows communication among more than two NVMe_Ports.

An FC-4 mapping layer uses the services provided by FC-FS-5 to perform the functions defined by the FC-4. The protocol is described in terms of the stream of FC IUs and Exchanges generated by a pair of NVMe_Ports that support the FC-4.

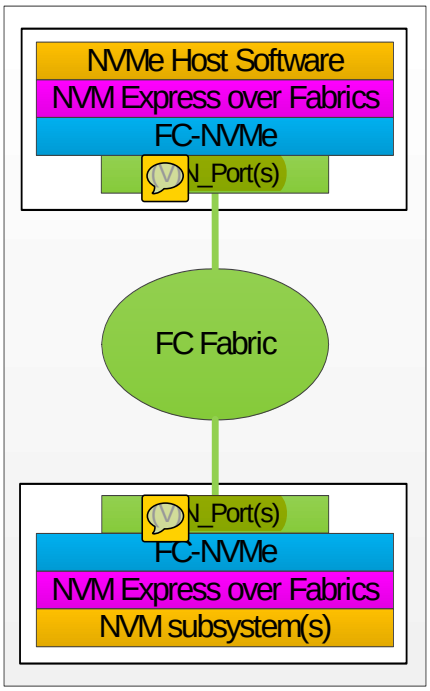
The detailed implementation that supports that stream is not defined by this standard. Originator and Responder NVMe_Ports are assumed to have a common service interface, for use by all FC-4s, that is similar in characteristics to the service interface defined in FC-FS-5.

This standard defines the following kinds of functional management:

a) device management;
b) Process Login and Process Logout management; and
c) link management.

The NVMe over FC protocol defines the mapping of NVM over Fabrics to the Fibre Channel interface (see FC-FS-5). Link control is performed by standard FC-FS-5 protocols. The I/O operation defined by NVMe over FC is mapped into a Fibre Channel Exchange. A Fibre Channel Exchange carrying information for an NVM Express over Fabrics I/O operation is an NVMe over FC Exchange. The request and response primitives of an I/O operation are mapped into Information Units (IUs) as specified in table 32 and table 33.

NVMe over FC protocol layers are shown in figure 1.



ure 1 – NVMe over FC protocol layers

The FC-NVMe protocol layer is specified in this standard, the NVM Express over Fabrics protocol layer is specified in the NVM Express over Fabrics specification, and the NVMe Host Software and NVM subsystem(s) protocol layer is specified is in the NVM Express specification.

The NVMe over FC target device functional model is shown in figure 2.

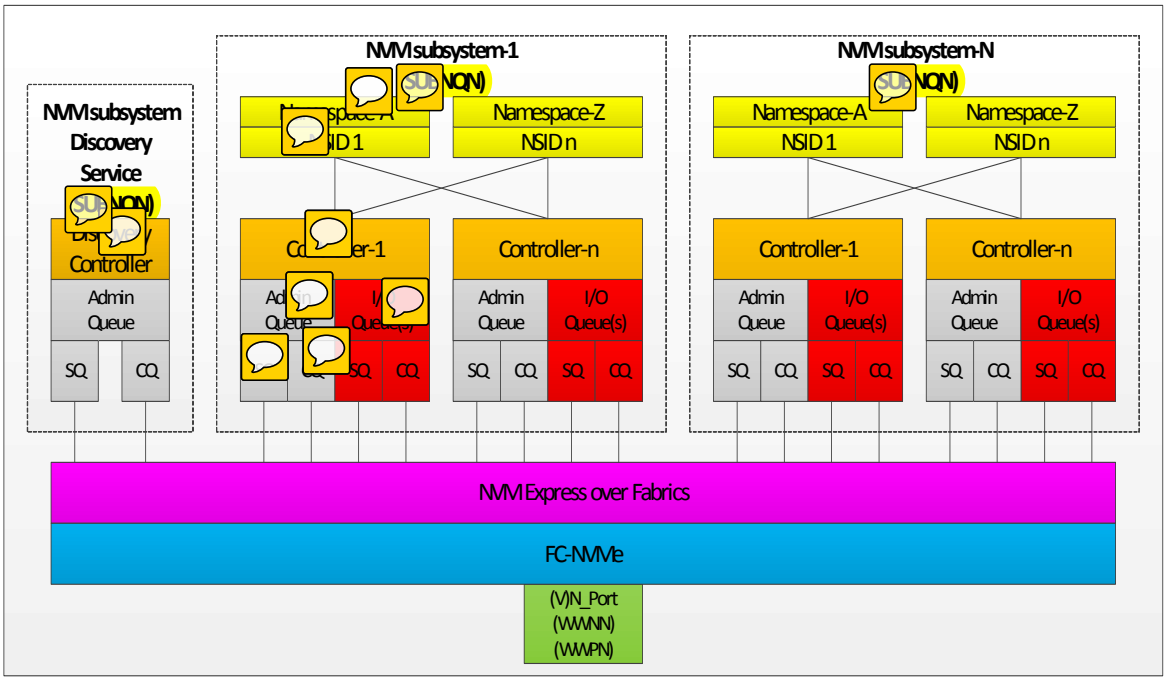


Figure 2 – NVMe over FC target device functional model

The mapping of NVM Express over Fabrics to NVMe over FC is specified in table 1.

Table 1 – NVM Express over Fabrics and NVMe over FC mapping

NVM Express over Fabrics	NVMe over FC equivalent	Reference
I/O operation	Exchange	FC-FS-5
SQL	Data Series	4.11

The number of Exchanges that may simultaneously be open between an initiator NVMe_Port and a target NVMe_Port is defined by the FC-FS-5 implementation. The architectural limit for this value is 65 535. ~~The maximum number of active Sequences that may simultaneously be open between an initiator NVMe_Port and a target NVMe_Port is restricted by the allowable range of values of the Sequence ID to 256 (see FC-FS-5).~~

4.2 NVMe over FC ports

A target NVMe_Port corresponds to a physical interface connecting a NVM subsystem to an FC Fabric. The target NVMe_Port is a NVM subsystem port for a particular NVM subsystem. The target NVMe_Port ~~may provide~~ Fabric connectivity for one or more NVM Subsystems. As NVM subsystem Port IDs (see table 40) are specific to a particular NVM subsystem and assigned by that NVM subsystem, a single target NVMe_Port providing connectivity for multiple NVM subsystems may be seen as multiple NVM subsystem ports with the same, or differing NVM subsystem Port ID values.

An initiator NVMe_Port corresponds to a physical interface connecting an NVMe host to an FC Fabric. The initiator NVMe_Port may provide Fabric connectivity for one or more NVMe hosts.

4.3 NVMe over FC association

An NVMe over FC association is an NVMe over FC transport abstraction for an exclusive communication relationship that is established between a particular NVMe host, connected via a particular initiator NVMe_Port, and a particular NVMe controller in an NVM subsystem connected via a particular target NVMe_Port. The association encompasses the controller, its state and properties, its Admin Queue, and all I/O Queues of that controller (see NVMe over Fabrics revision 1.0 and NVM Express revision 1.2.1).

The NVMe over FC association is created when the NVMe host makes a request of the NVMe over FC transport to establish the relationship to a particular controller on an NVM subsystem. The manner in which the NVMe host discovers possible NVM subsystems is outside the scope of this standard. On an FC Fabric, the request specifically identifies the initiator NVMe_Port and target NVMe_Port. The NVMe over FC transport initiates the creation of the association by transmitting a Create Association NVMe_LS request from the initiator NVMe_Port to the target NVMe_Port. The request identifies the NVMe host making the request by the HostNQN and Host ID values. The request identifies the NVM subsystem by its SUBNQN and the controller by its Controller ID value. The Controller ID value may identify a specific Controller ID or may request an available controller. If the target NVMe_Port and NVM subsystem allow the communication relationship to be created, the target NVMe_Port transmits a Create_Association accept payload to the initiator NVMe_Port. The accept payload contains an association identifier that shall be used by the NVMe over FC transport on the initiator NVMe_Port to refer to the NVMe over FC association in subsequent Fabric traffic transmitted to the target NVMe_Port. If the NVMe over FC association cannot be created, the target NVMe_Port shall transmit an NVMe_RJT to the initiator NVMe_Port with the reason code set to 03h (i.e., Logical error) and the reason code explanation set to an appropriate value.

An associated NVMe over Fabrics association is created by the first NVMe over Fabrics Connect command, issued on the association's Admin Queue connection, to create the Admin Queue. Refer to NVMe over Fabrics ~~revision 1.0~~ for additional requirements and behaviors of NVMe over Fabrics associations and Admin Queue creation.

Once established, the NVMe over FC association remains in place until:

a) the controller is shutdown;
b) a controller level reset;
c) a Keep Alive failure on the controller;
d) an NVMe over FC connection with the controller is terminated; or
e) an NVMe over FC transport event or clearing effect occurs that affects the communication between the initiator NVMe_Port and the target NVMe_Port (refer to the NVMe over Fabrics specification).

A communication reset may be relative to an individual association or all associations between the two NVMe_Ports. An NVMe over FC association may also be explicitly terminated by a Disconnect NVMe_LS request with the scope bit set to zero (i.e., the association) or with the scope bit set to one (i.e., a connection) and the Connection Identifier field value set to the Admin Queue's NVMe over FC Connection Identifier. If an initiator NVMe_Port or target NVMe_Port terminates an NVMe over FC association, the terminating NVMe_Port shall issue a Disconnect NVMe_LS request with the scope bit set to zero. All NVMe over FC compliant initiator NVMe_Ports shall be capable of transmitting a Disconnect NVMe_LS request with the scope bit set to zero and shall be capable of accepting and processing a Disconnect NVMe_LS request with the scope bit set to zero. ~~All NVMe over FC compliant target NVMe_Ports shall be capable of accepting and processing a Disconnect NVMe_LS~~

~~quest with the scope bit set to zero and shall be capable of transmitting a Disconnect NVMe_LS request with the scope bit set to zero.~~

If a NVMe over FC association is terminated, the NVMe over FC transport on the initiator NVMe_Port or target NVMe_Port shall implicitly terminate all Admin Queue and I/O Queue connections for the association. The controller shall be terminated and/or cleared according to the semantics defined in NVMe over Fabrics ~~revision 1.0~~ and NVM Express ~~revision 1.2.1~~.

If a target NVMe_Port receives a NVMe_LS request that does not correspond to an active NVMe over FC association, the target NVMe_Port shall not process the NVMe_LS request and shall transmit an NVMe_RJT with the reason code set to 40h (i.e., Invalid Association ID) and the reason code explanation set to 00h (i.e., No additional explanation).

4.4 NVMe over FC connection

An NVMe over FC connection is an NVMe over FC transport abstraction representing an NVMe Send Queue (SQ) and an NVMe Completion Queue (CQ) for a NVMe controller. The controller is identified by the NVMe over FC association. An NVMe over FC connection may correspond to the controller's Admin Queue or an I/O Queue on that controller.

An NVMe over FC connection corresponding to the Admin Queue is created simultaneously with the creation of the NVMe over FC association as part of the processing of the Create Association NVMe_LS request. The Create Association NVMe_LS request specifies the size of the Admin SQ, which is also the size of the Admin CQ, as well as the ERSP_Ratio mandating the periodic delivery of NVMe_ERSP IUs for completions. Successful creation of the NVMe over FC association shall also include creation of the resources for the NVMe over FC connection for the controller's Admin Queue (i.e., SQ and CQ). The Create Association accept payload specifies a Connection Identifier that shall be used by the NVMe over FC transport on the initiator NVMe_Port to refer to the controller's Admin Queue in subsequent Fabric traffic transmitted to the target NVMe_Port.

~~Created~~ NVMe controller Admin Queue is created by the first NVMe over Fabrics Connect command issued on the NVMe over FC connection for the controller's Admin Queue. See NVMe over Fabrics ~~revision 1.0~~ for additional requirements and behaviors of Admin Queue creation.

An NVMe over FC connection corresponding to an I/O Queue is created when the NVMe host makes a request of the NVMe over FC transport to establish the transport connection for an I/O Queue for a particular controller. The NVMe over FC transport initiates the creation of the transport connection by transmitting a Create I/O Connection NVMe_LS request from the initiator NVMe_Port to the target NVMe_Port. The request payload specifies the NVMe over FC association for the controller that the connection is to be established for as well as:

a) the I/O Queue ID it corresponds to;
b) the size of the Queue's SQ (which is also the size of its CQ); and
c) the ERSP_Ratio mandating the periodic delivery NVMe_ERSP IUs for completions.

If the target NVMe_Port and NVMe controller accept the request, the target NVMe_Port transmits a Create I/O Connection ~~accept payload~~ to the initiator NVMe_Port. The ~~accept~~ payload contains a Connection Identifier that shall be used by the NVMe over FC Transport on the initiator NVMe_Port to refer to the NVMe over FC connection in subsequent Fabric traffic transmitted to the target NVMe_Port (e.g., NVMe over FC I/O commands).

If a target NVMe_Port receives a Create I/O Connection NVMe_LS request with an Association Identifier that does not correspond to an active NVMe over FC association, the target NVMe_Port shall transmit an NVMe_RJT to the initiator NVMe_Port with the reason code set to 40h (i.e., Invalid

Association ID) and the reason code explanation set to 00h (i.e., No additional explanation). If the NVMe over FC connection cannot be created, the target NVMe_Port shall transmit an NVMe_RJT to the initiator NVMe_Port with the reason code set to 03h (i.e., Logical error) and the reason code explanation set to an appropriate value.

The first created NVMe controller I/O Queue is created by the first NVMe over Fabrics Connect command issued on the NVMe over FC connection for the controller's I/O Queue. See NVMe over Fabrics revision 1.0 for additional requirements and behaviors of I/O Queue creation.

Once established, the NVMe over FC connection remains in place until the NVMe over FC association is terminated, or a transport error occurs that causes loss of a message or loss of data in a way that the NVMe over FC transport cannot resolve or recover (see NVMe over Fabrics revision 1.0) (i.e., if an NVMe I/O operation is terminated with an ABTS-LS, as it potentially causes the loss of a SQE, CQE or data for an NVMe command, it shall cause the corresponding NVMe over FC connection to be terminated). An NVMe over FC connection may also be explicitly terminated by a Disconnect NVMe_LS request with the Scope bit set to one (i.e. a connection) and the Connection Identifier field set to the value for the NVMe over FC connection to be terminated. If an initiator NVMe_Port or target NVMe_Port terminates an NVMe over FC connection, the terminating NVMe_Port shall issue a Disconnect NVMe_LS request with the Scope set to one and the Connection Identifier field set to the value for the terminated NVMe over FC connection. All NVMe over FC compliant initiator NVMe_Ports shall be capable of transmitting a Disconnect NVMe_LS request with the Scope set to one and shall be capable of accepting and processing a Disconnect NVMe_LS request with the Scope set to one. All NVMe over FC compliant target NVMe_Ports shall be capable of accepting and processing a Disconnect NVMe_LS request with the Scope set to one and shall be capable of transmitting a Disconnect NVMe_LS request with the Scope set to one.

The termination of an NVMe over FC connection implicitly terminates the NVMe over Fabrics association with the controller thus implicitly terminating the controller's NVMe over FC association (see NVMe over Fabrics revision 1.0).

If an NVMe over FC connection is terminated, as the connection represents the NVMe Queue, the termination of the Queue also causes all outstanding NVM commands on the Queue to be implicitly terminated (see NVM Express revision 1.2.1). Thus, the termination of the NVMe over FC connection shall cause the NVMe over FC transport on the initiator NVMe_Port or target NVMe_Port to implicitly terminate all outstanding NVMe over FC I/Os that are associated with the NVMe over FC connection. For each outstanding NVMe over FC I/O operation on the connection, the initiator NVMe_Port shall transmit an ABTS-LS to terminate the Exchange for the I/O operation. If an NVMe over FC connection is terminated on a target NVMe_Port, the target NVMe_Port shall, for each outstanding NVMe over FC I/O operation on that connection, transmit an ABTS-LS to terminate the Exchange for the I/O operation.

If a target NVMe_Port receives an NVMe_LS request with a Connection Identifier that does not correspond to an active NVMe over FC connection, the target NVMe_Port shall not process the NVMe_LS. If an NVMe_LS request with a Connection Identifier for an inactive NVMe over FC connection is received, the target NVMe_Port shall transmit an NVMe_RJT with the reason code set to 41h (i.e., Invalid Connection ID) and the reason code explanation set to 00h (i.e., No additional explanation). If an NVMe IU with a Connection Identifier for an inactive NVMe over FC connection is received, the target NVMe_Port should transmit an ABTS-LS for the corresponding Exchange.

4.5 NVMe over FC I/O operations

When an NVMe host submits a command for processing by the controller, the command is submitted as a Submission Queue Entry (SQE) and an associated Scatter/Gather List to the NVMe over Fabrics layer which then submits the command to the NVMe over FC transport. The NVMe over FC

transport references the controller's NVMe over FC association and NVMe over FC connection for the SQ that the command is specific to, and delivers the command to the initiator NVMe_Port.

The initiator NVMe_Port begins the NVMe over FC I/O operation by allocating an Exchange resource and associating the NVMe command in the SQE to the Exchange. All NVMe IUs for the NVMe command shall be transmitted as part of the Exchange. The initiator NVMe_Port creates a NVMe_CMND IU for the Exchange. The NVMe_CMND IU payload conveys a single SQE (i.e. NVMe command) from the NVMe host to the NVMe controller via the NVMe over FC Connection (i.e., SQ). The NVMe_CMND IU specifies the Connection Identifier for the NVMe over FC connection, this NVMe controller and its Queue ID that the SQE is being submitted to. The NVMe_CMND IU also contains all the information necessary for the processing of the one command, including the local storage address and characteristics of data to be transferred by the command. ~~NVMe over FC Transport then performs the following actions using FC-FS-5 services to perform the command:~~

The initiator NVMe_Port transmits the NVMe_CMND IU payload to start the NVMe over FC I/O operation. The Exchange that is started is identified by its fully qualified Exchange identifier (FCXID) during the remainder of the NVMe over FC I/O operation and is used only for the IUs associated with that NVMe over FC I/O operation.

~~If the controller has interpreted the command and has determined that~~ a write operation is required, then the target NVMe_Port transmits a data descriptor IU containing the NVMe_XFER_RDY IU payload to the initiator NVMe_Port indicating which portion of the data is to be transferred. The initiator NVMe_Port then transmits a solicited data IU to the target NVMe_Port containing the NVMe_DATA IU payload requested by the NVMe_XFER_RDY IU. Data delivery requests containing NVMe_XFER_RDY IU and returning NVMe_DATA IU payloads continue until the data transfer requested by the command is complete. One NVMe_DATA IU shall follow each NVMe_XFER_RDY IU. If the initiator NVMe_Port and target NVMe_Port have negotiated to disable the initial NVMe_XFER_RDY IU, then a first burst may be transferred (see 4.5).

~~If the controller has interpreted the command and has determined that~~ a read operation is required, then the target NVMe_Port transmits a solicited data IU containing the NVMe_DATA IU payload to the initiator NVMe_Port. ~~Data deliveries containing~~ NVMe_DATA IU payloads continue until all data described by the command is transferred.

After ~~all the data has been transferred and command processing is completed~~, the target NVMe_Port transmits a ~~response IU~~. The response IU conveys an NVMe CQE which indicates the completion status of the NVMe command and in some cases, possible completion payload (see NVMe over Fabrics ~~revision 1.0~~ and NVM Express ~~revision 1.2.1~~). The response IU shall be a NVMe_RSP or NVMe_ERSP IU payload. The NVMe_ERSP IU payload contains a count of the amount of data transferred by the NVMe over FC transport as well as a CQE containing the completion status of the NVMe command and possible completion payload. The NVMe_RSP IU is used to convey an implied CQE with successful command completion with any additional completion data of zeros, as well as full transfer of all requested data for the NVMe command. The NVMe over FC transport on the initiator NVMe_Port shall convert the NVMe_RSP IU into a CQE with successful command status and zeros for any completion data.

If an error was detected in the processing of any of the command NVMe IUs, including the response IU, the initiator NVMe_Port shall terminate the NVMe over FC connection that the command was issued to. If the Exchange for the command is still open the initiator NVMe_Port shall transmit an ABTS-LS.

If no error was detected in the processing of the command's NVMe IUs, including the response IU, and the target NVMe_Port requested confirmed completion (see 4.10), then the initiator NVMe_Port shall transmit an NVMe_CONF IU with the LS bit set to one. If no error was detected in the processing

of any of the command NVMe IUs, including the response IU, and the Exchange was terminated either by the reception of the response IU with the LS bit set to one or by transmission of the NVMe_CONF IU with the LS bit set to one, then the initiator NVMe_Port shall deliver the CQE received or implied by the response IU to the NVMe CQ associated with the NVMe over FC connection that was specified in the NVMe_CMND IU.

~~NVMe over FC takes full advantage of the multiplexing and shared bandwidth capabilities provided by Fibre Channel classes of service. The protocol is designed to operate with Class 3 service and to provide options for reliable error detection and error recovery.~~

The NVMe initiator port function may exist in any NVMe_Port and the NVMe target port function may exist in any NVMe_Port. For NVMe over FC I/O operations between a host and a NVM subsystem, the host typically takes on the NVMe initiator port role and the NVM subsystem typically takes on the NVMe target port role.

4.6 First burst

If both the initiator NVMe_Port and target NVMe_Port support ~~a First Burst Supported bit (see table 8) value of one~~, then the initiator NVMe_Port may choose to perform write operations by sending the first NVMe_DATA IU without a preceding NVMe_XFER_RDY IU. The target NVMe_Port may accept or discard the first NVMe_Data IU. If the target NVMe_Port accepts the first NVMe_DATA IU, then the target NVMe_Port shall use NVMe_XFER_RDY IU(s) to request the transfer of any remaining write data ~~if any~~. If the target NVMe_Port discards the first NVMe_DATA IU, then the target NVMe_Port shall use NVMe_XFER_RDY IU(s) to request retransmission of write data originally sent in the first NVMe_DATA IU as well as for any remaining write data ~~if any~~. The initiator NVMe_Port shall support ~~retransmission of~~ write data originally sent in the first NVMe_DATA IU.

If either the initiator NVMe_Port or target NVMe_Port require the use of NVMe_XFER_RDY IUs during write operations (i.e., First Burst Supported bit is set to zero), then the initiator NVMe_Port shall not send a ~~first~~ NVMe_DATA IU without a preceding NVMe_XFER_RDY IU, and the target NVMe_Port shall transmit NVMe_XFER_RDY IU(s) requesting each NVMe_DATA IU(s) to perform the write data transfer.

4.7 In-order delivery requirements and behavior

4.7.1 Overview

NVMe_Ports and Fabrics shall provide in-order delivery of frames in an Exchange.

Some NVMe commands are required to be processed in the order they were sent by the initiator (i.e., fused operations (see NVM Express ~~rev. 1.2.1~~)). To allow these commands to be processed in the order they were sent, in the case that the order was not maintained by the Fabric, the Command Sequence Number is used (see 4.7.2).

For NVMe responses, all NVMe_ERSP IU responses are required to be processed in the order that they were sent. To allow the NVMe_ERSP IU to be processed in order, in the case that the order was not maintained by the Fabric, the Response Sequence Number is used (see 4.7.3).

There is no ordering requirement for the NVMe_RSP IU. The NVMe_RSP IU shall be processed by the NVMe_Port as soon as it is received. For an NVMe_RSP, a CQE shall be generated as specified in 4.8.2.

4.7.2 Command Sequence Number (CSN)

The Command Sequence Number is a four byte unsigned integer that starts at the reset value of zero and shall be incremented by one for each command. Separate increment counters are maintained for each NVMe connection. After the number of transmitted commands causes the integer to increment to 4294967295, the integer wraps back to a value of zero. The following rules specify how to use the CSN to determine that each command has been properly received and processed:

a) the CSN shall be equal to zero for the first NVMe_CMND IU for each NVMe connection and shall be incremented by one for each subsequent command;
b) the CSN shall wrap from 4294967295 to zero;
c) the initiator NVMe_Port shall not transmit the same CSN again until delivery of the first NVMe_CMND IU transmitted with that CSN has been confirmed by receipt of:
   A) an NVMe_XFER_RDY IU;
   B) the first Data frame of an NVMe_DATA IU;
   C) an NVMe_RSP IU or NVMe_ERSP IU; or
   D) an ABTS.

Editor's Note: Need to think about these rules in light of possible longer lived exchanges (i.e., Async Event or Abort).

d) the target NVMe_Port shall use the CSN to order any commands that are required to be in-order (i.e., fused operations (see NVM Express ~~rev. 1.2.1~~)).

4.7.3 Response Sequence Number (RSN)

The Response Sequence Number is a four byte unsigned integer that starts at the reset value of zero and shall be incremented by one for each NVMe_ERSP IU sent. Separate increment counters are maintained for each NVMe connection. After the number of transmitted NVMe_ERSP IUs causes the integer to increment to 4294967295, the integer wraps back to a value of zero. The following rules specify how to use the RSN to determine that each NVMe_ERSP IU has been properly received and processed:

a) the RSN shall be equal to zero for the first NVMe_ERSP IU for each NVMe connection and shall be incremented by one for each subsequent NVMe_ERSP IU;
b) the RSN shall wrap from 4294967295 to zero; and
c) the initiator NVMe_Port shall use the RSN to order all NVMe_ERSP IUs that are received.

4.8 NVMe_RSP IU response rules

4.8.1 Overview

The NVMe_RSP IU may be used by a target NVMe_Port to optimize data handling. If the NVMe_RSP IU is used:

a) an NVMe_ERSP IU shall be sent by the target NVMe_Port for every n responses, where n is specified by the NVMe_ERSP Ratio field value (see 8.2.4). This allows the SQHD to be updated in a regular fashion;
b) an NVMe_ERSP IU shall be sent by the target NVMe_Port if the SQ is 90% or more full. This allows the SQHD to be updated such that the SQ is not overwritten;
c) an NVMe_ERSP IU shall be sent by the target NVMe_Port for responses to aborted command;
d) an NVMe_ERSP IU shall be sent by the target NVMe_Port if the response capsule CQE contains a non-zero value in any location other than Command ID (CID) (bytes 13:12) and SQ Head Pointer (SQHD) (bytes 09:08) (i.e., ~~non-good completion status~~);

e) an NVMe_ERSP shall be sent by the target NVMe_Port if the Transferred Data Length field value is not equal to the NVMe_CMND IU Data Length field value; and
f) an NVMe_ERSP may be sent by the target NVMe_Port for any other reason when deemed necessary by the target NVMe_Port.

4.8.2 NVMe_RSP CQE fields

For commands completed by an NVMe_RSP IU, the NVMe CQE shall be generated as follows:

a) SQHD set to value received from the last SQHD sent to the NVMe layer;
b) Command_ID set to value sent in original NVMe SQE; and
c) all other fields set to zero.

4.9 NVMe_ERSP IU response rules

If a command is completed by an NVMe_ERSP IU with a Status Code field value of 00h (i.e., NONE) and the Transferred Data Length field value is equal to the initiator NVMe_Port's transfer byte count, then the command shall be completed by delivery of the CQE contained within the NVMe_ERSP IU to the NVMe layer.

If a command is completed by an NVMe_ERSP IU with a Status Code field value other than 00h ~~(i.e., NONE)~~ or with a Transferred Data Length field value that is not equal to the initiator NVMe_Port's transfer byte count, then the NVMe over FC transport shall not deliver the CQE to the NVMe layer, but shall communicate the detected error to the NVMe layer. The manner in which the NVMe over FC transport communicates the error to the NVMe layer, and the resulting recovery actions, is outside the scope of this standard.

4.10 Confirmed completion of NVMe over FC I/O operations

Some implementations require an acknowledgment of successful delivery of NVMe_RSP IU or NVMe_ERSP IU information (i.e., confirmed completion). Such an acknowledgment is provided by requesting an NVMe_CONF IU. The Confirmed Completion Supported bits in the PRLI ELS request NVMe over FC Service Parameter page (see 6.3.3) and PRLI ELS accept NVMe over FC Service Parameter page (see 6.3.4) are used to negotiate the support for confirmed completion.

If an initiator NVMe_Port and a target NVMe_Port both support confirmed completion, then a target NVMe_Port may request an NVMe_CONF IU by not setting the Last_Sequence bit to one (see FC-FS-5) in the last frame of an NVMe_RSP IU or NVMe_ERSP IU. Upon detecting the NVMe_CONF IU request, the initiator NVMe_Port shall transmit an NVMe_CONF IU to the target NVMe_Port, indicating to the target NVMe_Port that the NVMe_RSP IU or NVMe_ERSP IU has been received by the initiator NVMe_Port.

Confirmed completion provides a confirmation that the initiator NVMe_Port and the target NVMe_Port both agree upon the state of a state dependent device.

Confirmed completion may assist NVMe initiator devices and NVMe target devices in many environments. Examples include:

a) confirmed completion may be used to confirm that an initiator NVMe_Port has received an NVMe_ERSP IU reporting a non-Successful Completion status, together with accompanying additional data. Upon receiving the NVMe_CONF IU, the target NVMe_Port may discard its copy of the error condition data;
b) confirmed completion may be used to confirm that a command has been completed and that the completion information has been successfully transferred to the initiator NVMe_Port. That

allows subsequent queued state dependent operations to be performed, since the NVMe_CONF IU confirms that the NVMe_RSP IU or NVMe_ERSP IU has been received by the initiator NVMe_Port; and

c) confirmed completion may be used to confirm that an initiator NVMe_Port has received the NVMe_RSP IU or NVMe_ERSP IU for target NVMe_Ports that require state dependent synchronization with initiator NVMe_Ports.

4.11 Data transfer

4.11.1 Overview

NVMe over Fabrics (see NVMe over Fabrics revision 1.0) specifies two types of data transfers:

a) in-capsule data; and
b) SGL data.

In-capsule data is data transferred within the capsule. SGL data is data specified by a list of memory regions. All data transfers for NVMe over FC shall be converted to Data Series transfers (see figure 3). In order to map NVM Express fabric data transfers onto NVMe over FC this subclause specifies rules for the following types of NVMe over Fabrics data transfers:

a) in-capsule data;
b) SGL data for writes; and
c) SGL data for reads.

4.11.2 In-capsule data

In-capsule data shall be transferred in a Data Series. For data and metadata, the capsule contains pointers to the offset into the Data Series. The format of these pointers are specified in NVM Express (see NVM Express revision 1.2.1).

4.11.3 SGL data

4.11.3.1 Overview

NVMe over Fabrics defines a mechanism for transmitting SGLs across a Fabric. Fibre Channel does not send SGLs across the Fabric (e.g., such transmission requires an RDMA mechanism, see NVMe over Fabrics revision 1.0), thus SGLs shall be converted to data sent within a Data Series for transmission across a Fibre Channel Fabric.

4.11.3.2 SGL mapping

An NVMe SGL is a list of memory regions to be gathered by the receiving NVMe controller. In order for data referenced by an SGL to be transferred via NVMe over FC ~~the following shall occur:~~

a) on a write, the data pointed to by the SGL shall be placed into a Data Series;
b) on a read, the data shall be placed into a Data Series by the NVMe controller; and
c) for both read and write, the SGL data field within the SQE shall be replaced by an offset into the Data Series and the length of the data (see 4.11.3.3).

An SGL example is shown in figure 3.

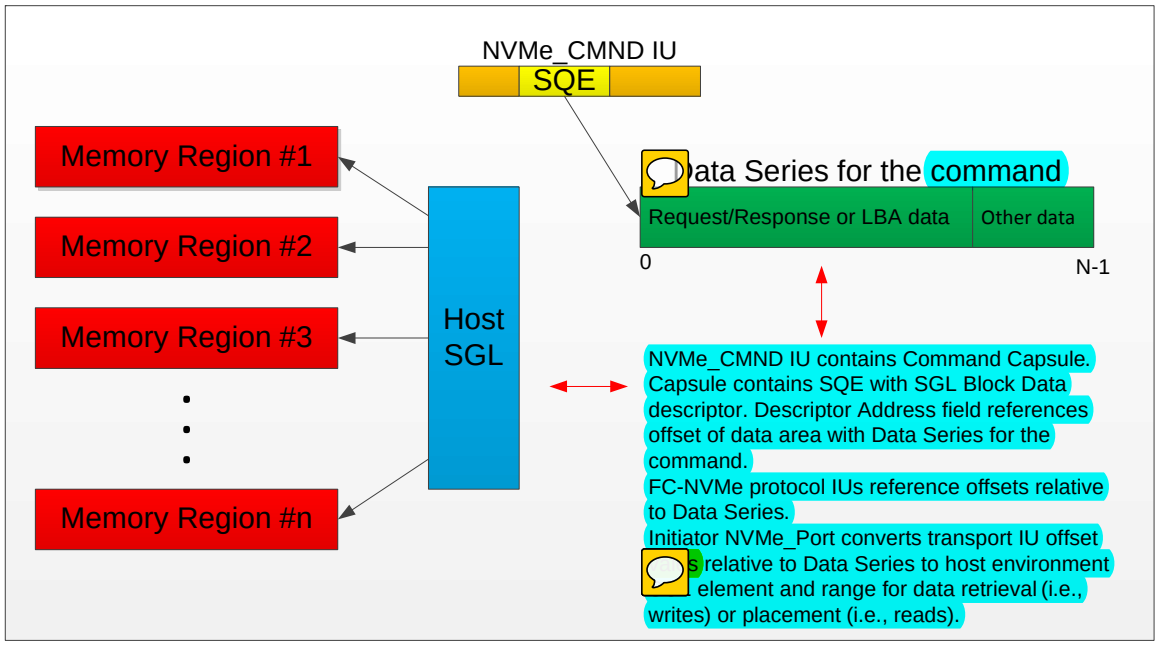


Figure 3 – SGL example

4.11.3.3 SGL entry format

For transmission the SGL within the SQE shall be (see NVMe over Fabrics revision 1.0):

a) the SGL Descriptor Type field shall be set to 0h;
b) the SGL Descriptor Sub Type field shall be set to 0h;
c) the Address field of the SGL Data Block descriptor shall be set to zero; and
d) the Length field of the SGL Data Block descriptor shall contain the length of the data in the Data Series.

4.12 Discovery of NVMe over FC capabilities

A number of NVMe over FC capabilities require the knowledge and agreement of both the target NVMe_Port and the initiator NVMe_Port that such capabilities may or shall be used. Table 2 provides references to the discovery process for each of the NVMe over FC capabilities.

Table 2 – Discovery of NVMe over FC capabilities

Capability	Discovery mechanism	Reference
Initiator NVMe_Port	Process Login	6.3
Target NVMe_Port	Process Login	6.3
Discovery Service	Process Login	6.3
Confirmed Completion Supported	Process Login	6.3
First Burst Supported	Process Login	6.3

4.13 Clearing effects of NVMe over FC, FC-FS-5, and FC-LS-3 actions

Table 3 and table 4 summarize the clearing effects resulting from Fibre Channel link actions and NVMe over FC operations, respectively. The clearing effects are applicable only to Sequences and Exchanges associated with NVMe over FC actions. Sequences and Exchanges associated with other actions follow rules specified in FC-FS-5 and other relevant protocol standards. Rows indicating a clearing effect for all initiator NVMe_Ports have the specified clearing effect on all initiator NVMe_Ports, regardless of the link that attaches the initiator NVMe_Port to the target NVMe_Port.

Clearing effects of link related actions are specified in table 3.

Table 3 – Clearing effects of link related actions

Clearing effect	FC link action					
	Target Port Cycle	LOGO ELS[b], PLOGI ELS	PRLI ELS PRLO ELS[b],	TPRLO ELS[a]	ABTS-LS	ABTS (Sequence)
PLOGI ELS parameters set to default values (see FC-LS-3)						
For all logged-in initiator NVMe_Ports		N	N	N	N	N
Only for initiator NVMe_Port associated with the action		N	N	N	N	N
Active NVMe over FC Associations terminated						
For all initiator NVMe_Ports	Y	N	N	Y	N	N
Only for initiator NVMe_Port associated with the action	-	Y	Y	-	N	N
Only for NVMe over FC Association associated with the action	-	-	-	-	Y	N
Active NVMe over FC Connections terminated						
For all initiator NVMe_Ports	Y	N	N	Y	N	N
Only for initiator NVMe_Port associated with the action	-	Y	Y	-	N	N
Only for NVMe over FC Association associated with the action	-	-	-	-	Y	N
Only for NVMe over FC Connection associated with the action	-	-	-	-	-	N
Open NVMe over FC Exchanges terminated						
For all initiator NVMe_Ports	Y	N	N	Y	N	N
Only for initiator NVMe_Port associated with the action	-	Y	Y	-	N	N
Only for NVMe over FC Association associated with the action	-	-	-	-	Y	N
Only for NVMe over FC Connection associated with the action	-	-	-	-	-	N
Only for NVMe over FC Exchange associated with ABTS	-	-	-	-	-	N
NVMe over FC Sequence associated with ABTS terminated	-	-	-	-	-	Y
On BB_Credit_CNT set to login value (see FC-FS-4)						
For all Logged-In NL_Ports	Y	N	N	N	N	N
For transmitting NL_Port only	-	Y	N	N	N	N
Process Login parameters cleared[c]						
For all logged-in initiator NVMe_Ports	Y	N	N	Y	N	N
Only for NVMe_Port associated with the action	-	Y		-	N	N
CSN set to zero						
For all initiator NVMe_Ports	Y	N	N	Y	N	N
Only for initiator NVMe_Port associated with the action	-	Y	Y	-	N	N

Table 3 – Clearing effects of link related actions (Continued)

Clearing effect	FC link action					
	Target Power Cycle	LOGO ELS[b], PLOGI ELS	PRLI ELS PRLO ELS[b],	TPRLO ELS[a]	ABTS-LS	ABTS (Sequence)
Key: "Y" indicates the clearing effect upon successful completion of the specified action. "N" indicates the clearing effect is not performed by the specified action. "-" indicates the clearing effect is not applicable. a) For a TPRLO ELS, the actions listed shall be performed when the GLOBAL bit is set to one. If the GLOBAL bit is set to zero, then the actions listed under PRLI ELS/PRLO ELS shall be performed for the designated initiator NVMe_Port. See FC-FS-5. b) Logout and Process Logout may be either implicit or explicit. Implicit logout and Process Logout are specified in FC-FS-5. c) A target NVMe_Port should transmit a PRLO ELS to all logged-in initiator NVMe_Ports that are logged out as a result of processing a TPRLO ELS with the GLOBAL bit set to one. The PRLO ELS(s) may be transmitted before or after transmitting the LS_ACC for the TPRLO ELS.						

Clearing effects of initiator NVMe_Port actions are specified in table 4.

Table 4 – Clearing effects of initiator NVMe_Port actions

Clearing effect	Initiator NVMe_Port action			
	troller eset	DISCONNECT NVME_LS scope=0 (Association)	DISCONNECT NVME_LS scope=1 (Queue) &Q	DISCONNECT NVME_LS scope=1 (Queue) &Q ID !=0
PLOGI ELS parameters set to default values (see FC-LS-3)				
For all logged-in initiator NVMe_Ports	N	N	N	N
Only for initiator NVMe_Port associated with the action	N	N	N	N
Active NVMe over FC Associations terminated				
For all initiator NVMe_Ports	N	N	N	N
Only for initiator NVMe_Port associated with the action	N	N	N	N
Only for NVMe over FC Association associated with the action	Y	Y	Y	N
Active NVMe over FC Connections terminated				
For all initiator NVMe_Ports	N	N	N	N
Only for initiator NVMe_Port associated with the action	N	N	N	N
Only for NVMe over FC Association associated with the action	Y	Y	Y	N
Only for NVMe over FC Connection associated with the action	-	-	-	Y
Open NVMe over FC Exchanges terminated				
For all initiator NVMe_Ports	N	N	N	N
Only for initiator NVMe_Port associated with the action	N	N	N	N
Only for NVMe over FC Association associated with the action	Y	Y	Y	N
Only for NVMe over FC Connection associated with the action	-	-	-	Y
Open NVMe over FC Sequences Terminated				
For all initiator NVMe_Ports with open NVMe over FC Sequences	-	N	N	N
Only for initiator NVMe_Port associated with the action	-	N	N	N
Only for NVMe over FC Sequences associated with aborted NVMe over FC Exchanges	-	N	N	N

Table 4 – Clearing effects of initiator NVMe_Port actions (Continued)

Clearing effect	Initiator NVMe_Port action			
	Controller Reset	DISCONNECT NVME_LS scope=0 (Association)	DISCONNECT NVME_LS scope=1 (Queue) &Q ID =0	DISCONNECT NVME_LS scope=1 (Queue) &Q ID !=0
BB_Credit_CNT set to login value (see FC-FS-5) For all Logged-In NL_Ports For transmitting NL_Port only	N N	N N	N N	N N
Process Login parameters cleared For all logged-in initiator NVMe_Ports Only for NVMe_Port associated with the action				
set to zero For all initiator NVMe_Port Only for initiator NVMe_Port associated with the action				

Key:
"Y" indicates the clearing effect upon successful completion of the specified action.
"N" indicates the clearing effect is not performed by the specified action.
indicates the clearing effect is not applicable.

a) Exchanges are cleared internally within the target NVMe_Port, but open NVMe over FC Sequences shall be individually aborted by the initiator NVMe_Port using ABTS-LS. This has the effect of aborting the associated NVMe over FC Exchange.

4.14 Port Login/Logout

The N_Port Login (PLOGI) ELS is optionally used to establish the Fibre Channel operating parameters between any two Fibre Channel ports, including NVMe_Ports. Implicit login functions are allowed.

If a target NVMe_Port receives a PLOGI ELS request and it finds there are not enough login resources to complete the login, then the target NVMe_Port shall respond to the PLOGI ELS with LS_RJT and Reason Code "Unable to perform command request" and Reason Code Explanation "Insufficient resources to support Login" as defined in FC-LS-3. By means outside the scope of this standard, the target NVMe_Port may select another initiator NVMe_Port and release some login resources by performing an explicit logout of the other initiator NVMe_Port, thus freeing resources for a future PLOGI ELS.

4.15 Process Login and Process Logout

The Process Login (PRLI) ELS request is used to establish the NVMe over FC operating relationships between two NVMe_Ports (see 6.3). The Process Logout (PRLO) ELS request is used to de establish the NVMe over FC operating relationships between two NVMe_Ports (see 6.4). Implicit Process Login and Process Logout parameters may be defined for NVMe_Ports. Such definitions are outside the scope of this standard.

4.16 Link management

FC-FS-5 allows management protocols above the FC-FS-5 interface to perform link data functions. The standard primitive sequences, link management protocols, BLSs, and ELSs are used as required by NVMe over FC devices (see FC-FS-5 and FC-LS-3).

4.17 NVMe over FC addressing and Exchange identification

The address of each NVMe_Port is defined by its address identifier as described in FC-FS-5.

Each NVMe over FC association is identified by the Association Identifier created by a successful Create Association NVMe_LS request. The Association Identifier is valid as long as the NVMe over FC association is active.

Each NVMe over FC connection is identified by the Connection Identifier created by a successful Create Association NVMe_LS request or Create I/O Connection NVMe_LS request. The Connection Identifier is valid as long as the NVMe over FC connection is active.

Each NVMe over FC I/O operation is identified by the NVMe over FC I/O operation's fully qualified exchange identifier (FQXID). The FQXID is composed of the initiator port identifier, the target port identifier, the OX_ID field value, and the RX_ID field value. Other definitions of FQXID are outside the scope of this standard. The method used to identify NVMe over FC I/O operations internal to the host and the controller is not defined by this standard.

Namespace Identifiers are contained in the NVM Submission Queue Entry field of NVMe_CMD IUs. Subsequent identification of the NVMe over FC I/O operation and the Exchange that carries the protocol interactions for the NVMe over FC I/O operation uses the FQXID.

4.18 Use of Worldwide_Names

As specified in FC-FS-5, each Fibre Channel node shall have a Node_Name that is a Worldwide_Name and each Fibre Channel port shall have an N_Port_Name that is a Worldwide_Name. The Worldwide_Name shall be unique within the FC-NVMe interaction space using one of the formats defined by FC-FS-5. Each target NVMe_Port and its associated NVM subsystems have knowledge of the N_Port_Name of each initiator NVMe_Port through the Fibre Channel login process.

The FC-NVMe interaction space is the set of Fibre Channel ports, devices, and Fabrics that are connected by a Fibre Channel administrative/management entity, or are accessible by a common instance of a Fibre Channel administrative tool or tools.

NOTE 1 – WWN uniqueness between separate FC-NVMe interaction spaces is outside the scope of this standard.

The Worldwide_Name for the NVMe_Port shall be different from the Worldwide_Name for the node (i.e., the N_Port_Name shall be different than the Node_Name).

5 FC-FS-5 Frame_Header

The format of the NVMe over FC Frame_Header is specified in table 5.

Table 5 – NVMe over FC Frame_Header

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	R_CTL	D_ID		
1	CS_CTL/Priority	S_ID		
2	TYPE	F_CTL		
3	SEQ_ID	DF_CTL	SEQ_CNT	
4	OX_ID		RX_ID	
5	Parameter			

All fields in the NVMe over FC Frame_Header use the standard FC-FS-5 definitions. The following explanations of the fields provide information about the use of those fields to implement NVMe over FC functionality.

R_CTL: The R_CTL field is subdivided into a ROUTING field and an INFORMATION field (see FC-FS-5). The ROUTING field shall be set to 0h (i.e., Device_Data) and the INFORMATION field shall be set to the value specified in table 32 and table 33.

D_ID: The value in the D_ID field is the D_ID of the frame. For NVMe over FC FC-4 Device_Data frames, the D_ID transmitted by the Exchange Originator is the address identifier of the target NVMe_Port. The D_ID transmitted by the Exchange Responder is the address identifier of the initiator NVMe_Port.

CS_CTL/Priority: The values in the CS_CTL/Priority field are defined by FC-FS-5 for class specific control information or priority and do not interact with the NVMe over FC protocol.

S_ID: The value in the S_ID field is the S_ID of the frame. For NVMe over FC FC-4 Device_Data frames, the S_ID transmitted by the Exchange Originator is the address identifier of the initiator NVMe_Port. The S_ID transmitted by the Exchange Responder is the address identifier of the target NVMe_Port.

TYPE: The value in the TYPE field shall be set to:

a) 08h (i.e., Fibre Channel Protocol) (see FC-FS-5) for all frames of NVMe over FC Exchanges using IUs specified in table 32 and table 33; or
b) 28h (i.e., NVMe over Fibre Channel) (see FC-FS-5) for all other frames.

6 NVMe over FC Link Services

6.1 Overview of Link Service requirements

The NVMe over FC link-level protocol includes the Basic Link Services (see FC-FS-5) and Extended Link Services (see FC-LS-3). The protocol also includes the PRLI ELS and PRLO ELS specified in FC-LS-3, and the PRLI NVMe over FC Service Parameter pages specified in 6.3.

Link-level protocols are used to configure the FC environment, including the establishment of configuration information and address information. NVMe over FC devices introduced into a configuration or modifications in the addressing or routing of the configuration may require the login and discovery procedures to be performed again.

6.2 Overview of Process Login and Process Logout

The PRLI ELS is used to exchange Process Login service parameters between an initiator NVMe_Port and a target NVMe_Port, and is not used to establish logical image pairs.

~~Since explicit login may be established by configuration conventions outside the scope of this standard, Process Login is optional except in the case where an initiator NVMe_Port is not using implicit login and is operating in a point-to-point topology. In this case, the initiator NVMe_Port shall always transmit an explicit PRLI ELS.~~

~~NOTE 2 – The requirement to transmit a PRLI ELS for an initiator NVMe_Port that is not using implicit login and operating in a point-to-point topology is to remove a deadlock condition that occurs when the target NVMe_Port N_Port_Name is larger than the initiator NVMe_Port N_Port_Name. In this case the target NVMe_Port PLOGI ELS request is processed, but the target NVMe_Port is prohibited from transmitting a PRLI ELS. If the initiator NVMe_Port does not transmit a PRLI ELS, then a deadlock occurs.~~

PRLI ELS requests shall only be initiated by devices having the initiator NVMe_Port capability. Devices having only target NVMe_Port capability shall not perform a PRLI ELS request.

An initiator NVMe_Port shall have successfully completed Process Login with a target NVMe_Port before any NVMe over FC IUs are exchanged. ~~An implicit Process Login may be performed by methods outside the scope of this standard.~~ Any NVMe over FC IUs received by a target NVMe_Port from an Nx_Port that has not successfully completed Process Login with that target NVMe_Port shall be discarded. In addition, a target NVMe_Port that receives an NVMe_CMND IU from an Nx_Port that it has successfully completed PLOGI ELS with, but has not successfully completed Process Login with that target NVMe_Port, shall discard the NVMe_CMND IU and respond with an explicit PRLO ELS (see 6.4).

The FC-4 Service Parameter pages for the NVMe over FC protocol are defined in 6.3.3 and 6.3.4.

Processing of a PRLI ELS or PRLO ELS request performs the clearing actions defined in table 3 and table 4.

6.3 PRLI ELS

6.3.1 Use of PRLI ELS

The PRLI ELS request is transmitted from an initiator NVMe_Port to a target NVMe_Port to exchange Process Login service parameters (see FC-LS-3).

Process Login is successfully completed only if the NVMe over FC devices have complementary initiator NVMe_Port and target NVMe_Port capabilities. Some capabilities require support by both the initiator NVMe_Port and target NVMe_Port before they may be used (see 6.3.3).

An accept response code indicating other than 'Request executed' (see 6.3.4 and FC-LS-3) shall be provided if the PRLI ELS NVMe over FC Service Parameter page is incorrect.

A Link Service Reject (LS_RJT) indicates that the PRLI ELS request is not supported or is incorrectly formatted.

The PRLI ELS common service parameters and accept response codes are defined in FC-LS-3.

6.3.2 New or repeated Process Login

After the completion of any new or repeated Process Login, all clearing actions specified in table 3 and table 4 shall be performed.

6.3.3 PRLI ELS request NVMe over FC Service Parameter page format

The NVMe over FC Service Parameter page for the PRLI ELS request is specified in table 6.

Table 6 – PRLI ELS request NVMe over FC Service Parameter page

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	TYPE Code	TYPE Code Extension	Common Information	
1	Reserved			
2	Reserved			
3	Service Parameter Information			
4	Reserved			

TYPE Code: Shall be set to 28h to indicate this Service Parameter page is defined for NVMe over Fibre Channel (see FC-FS-5).

TYPE Code Extension: Shall be set to 00h.

Common Information: The format of the Common Information field is specified in table 7.

Table 7 – Common Information field format

Bit	Description
15	Reserved
14	Reserved
13	Establish Image Pair: Shall be set to zero.
12 to 0	Reserved

Service Parameter Information: The format of the Service Parameter Information field is specified in table 8.

Table 8 – Service Parameter Information field format - request and accept

Bit	Description
31 to 8	Reserved
7	Confirmed Completion Supported: If set to one, then confirmed completion is supported (see 4.10). If set to zero, then confirmed completion is not supported.
6	Reserved
5	Initiator Function: If the Initiator Function bit is set to one, then the Originator or Responder is indicating it has the capability of operating as an initiator NVMe_Port. If the Initiator Function bit is set to zero, then the Originator or Responder does not have the capability of operating as an initiator NVMe_Port.
4	Target Function: If the Target Function bit is set to one, then the Originator or Responder is indicating that it has the capability of operating as a target NVMe_Port. If the Target Function bit is set to zero, then the Originator or Responder does not have the capability of operating as a target NVMe_Port. Both the Initiator Function bit and the Target Function bit may be set to one. If neither the Initiator Function bit nor the Target Function bit is set to one, then the service parameters for the NVMe over FC Service Parameter page are assumed to be invalid. A Responder receiving such an invalid NVMe over FC Service Parameter page shall notify the Originator with a PRLI ELS accept response code of 'Service Parameters are invalid'. An Originator receiving such an invalid NVMe over FC Service Parameter page shall not perform NVMe over FC protocol operations with the Responder.
3	Discovery Service: If the Discovery Service bit is set to one, then the Originator or Responder is indicating that it has the capability of operating as a Discovery Service as specified in NVMe over Fabrics. If the Discovery Service bit is set to zero, then the ~~Originator or~~ Responder does not have the capability of operating as a Discovery Service.
2 to 1	Reserved
0	First Burst Supported: If set to one, then first burst is supported (see clause 4.3). If set to zero, then first burst is not supported.

6.3.4 PRLI ELS accept NVMe over FC Service Parameter page format

The NVMe over FC Service Parameter page for the PRLI ELS accept is shown in table 9.

Table 9 – PRLI ELS accept NVMe over FC Service Parameter page

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	TYPE Code	TYPE Code Extension	Common Information	
1	Reserved			
2	Reserved			
3	Service Parameter Information			
4	Reserved		First Burst Size	

h the following exceptions, the service parameter definitions are identical fo the PRLI ELS request (see table 6) ac ep NVMe over FC Service Parameter pages.

Common Information: The format of the Common Information field is specified in table 10.

Table 10 – Common Information field format

Bit	Description
15	Reserved
14	Reserved
13	Establish Image Pair: Shall be set to zero.
12	Reserved
11 to 8	Response Code: The Response Code field is defined in FC-LS-3.
7 to 0	Reserved

Service Parameter Information: The format of the Service Parameter Information field is specified in table 8.

First Burst Size: If first burst is not supported (see table 8), then the First Burst Size field gnored.

If the First Burst Size field is set to zero, then there is no first burst size limit.

If the First Burst Size field is not set to zero and first burst is supported, then the First Burst Size field indicates the maximum number of bytes that shall be transmitted in the first NVMe_DATA IU sent from the initiator NVMe_Port to the target NVMe_Port.

The First Burst Size field value is expressed in increments of 512 bytes (i.e., a value of one means 512 bytes, two means 1024 bytes). Support for the First Burst Size field shall be implemented by all NVMe over FC devices.

data transmissions from the target NVMe_Port to the initiator NVMe_Port, the First Burst Size field is ignored.

6.4 PRLO ELS

The format for the PRLO ELS request and PRLO ELS accept is specified in FC-LS-3.

The PRLO ELS request is transmitted from an Originator NVMe_Port to a Responder NVMe_Port to request that a Process Logout be performed. If the Process Logout completes successfully, then all clearing actions specified in 4.13 shall be performed.

The PRLO ELS accept shall present a response NVMe over FC Service Parameter page for the request NVMe over FC Service Parameter page. It is not an error to perform Process Logout for a Process Login that does not exist.

A Link Service Reject (LS_RJT) indicates that the PRLO ELS request is invalid and not accepted.

After Process Logout, no further NVMe over FC communication is possible between those Nx_Ports.

The PRLO ELS accept response codes are defined in FC-LS-3.

7 FC-4 Name Server registration and objects

7.1 Overview of FC-4 specific objects for NVMe over FC

The Name Server for a Fibre Channel Fabric is specified in FC-GS-8. NVMe over FC specific objects are specified in this clause for use by the Name Server. FC-GS-8 provides complete descriptions of the operations that are performed to register objects with a Name Server and to query the Name Server for the value of the objects.

7.2 FC-4 TYPEs object

The FC-4 TYPEs object (see FC-GS-8) indicates a set of supported data structure type values for Device_Data and FC-4 Link_Data frames (see FC-FS-5).

An NVMe_Port shall register the NVMe over Fibre Channel TYPE (28h) with the Name Server using the RFT_ID request CT_IU. This registration shall precede registration of the FC-4 TYPE 28h FC-4 Features object.

7.3 FC-4 Features object

The FC-4 Features object (see FC-GS-8) defines a 4-bit field for each FC-4 TYPE code. The FC-4 Features object is a 32-word array of 4-bit values. The 4-bit FC-4 Features bits for NVMe over Fibre Channel TYPE 28h are inserted in bits 3 to 0 of word 5. The format of the 4-bit FC-4 Features bits for NVMe over Fibre Channel TYPE 28h is shown in table 11.

Table 11 – FC-4 Features bits for NVMe over FC

Bit	Description
3	Reserved
2	Discovery Service (see NVM Express over Fabrics) supported. If the Discovery Service bit is set to one, then the NVMe_Port is indicating that it has the capability of operating as a Discovery Service as specified in NVMe over Fabrics. If the Discovery Service bit is set to zero, then the NVMe_Port does not have the capability of operating as a Discovery Service as specified in NVMe over Fabrics.
1	NVMe over FC initiator function supported. If the NVMe over FC initiator function bit is set to one, then the NVMe_Port is indicating that it has the capability of operating as an NVMe over FC initiator. If the NVMe over FC initiator function bit is set to zero, then the NVMe_Port does not have the capability of operating as an NVMe over FC initiator.
0	NVMe over FC target function supported. If the NVMe over FC target function bit is set to one, then the NVMe_Port is indicating that it has the capability of operating as an NVMe over FC target. If the NVMe over FC target function bit is set to zero, then the NVMe_Port does not have the capability of operating as an NVMe over FC target.

8 NVMe FC-4 Link Services

8.1 Overview

FC-4 Link Service functionality is specified in FC-LS-3. For NVMe FC-4 Link Services, the Frame_Header fields (see 5) shall be set as follows:

a) R_CTL Routing field (word 0, bits 31-28) shall be set to 0011b (i.e., an FC-4 Link_Data frame);
b) the TYPE field shall be set to 28h (i.e., FC-NVMe FC-4 Link Service frame); and
c) the R_CTL Information field (word 0, bits 27-24) shall be set to 0010b (i.e., unsolicited control) for request Sequences and 0011b (i.e., solicited control) for response Sequences.

An NVMe FC-4 Link Service request and response shall be a single frame Sequence unless otherwise specified.

The NVMe_LS requests and responses are specified in table 12.

Table 12 – NVMe_LS requests and responses

Value (Bits 31-24)	Description	Abbr.	Reference
01h	NVMe_LS reject	NVMe_RJT	8.3
02h	NVMe_LS accept	NVMe_ACC	8.4
03h	Create Association	CASS	8.5
04h	Create I/O Connection	CIOC	8.6
05h	Disconnect	DISC	8.7
All others	Reserved		

8.2 NVMe Link Service descriptors

8.2.1 Overview

The NVMe_LS descriptors are specified in table 13.

Table 13 – NVMe_LS descriptors

Tag value	Description	Reference
0000 0000h	Reserved	
0000 0001h	Link Service Request Information	8.2.2
0000 0002h	Reject	8.2.3
0000 0003h	Create Association Command	8.2.4
0000 0004h	Create I/O Connection Command	8.2.5
0000 0005h	Disconnect Command	8.2.6
0000 0006h	Connection Identifier	8.2.7
0000 0007h	Association Identifier	8.2.8
All others	Reserved	

8.2.2 Link Service Request Information descriptor

The format of the Link Service Information descriptor is specified in table 14.

Table 14 – Link Service Request Information descriptor

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	Descriptor tag = 0000 0001h			
1	Descriptor length			
2	Request payload word 0			
3	Reserved			

Descriptor length: The Descriptor length field contains the length in bytes of the following payload.

Request payload word 0 value: contains the value of word 0 (i.e., the NVMe LS word that contains the command code) specified in the associated NVMe Link Service request.

8.2.3 Reject descriptor

The format of the Reject descriptor is specified in table 15.

Table 15 – Reject descriptor

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	Descriptor tag = 0000 0002h			
1	Descriptor length			
2	Reserved	reason code	reason code explanation	vendor specific
3	Reserved			

Descriptor length: The Descriptor length field contains the length in bytes of the following payload.

reason code: contains a value specified in table 16.

Table 16 – NVMe_LS reason codes

Value	Description	
01h	Invalid NVMe_LS command code	
03h	Logical error	
09h	Unable to perform command request	
0Bh	Command not supported	
0Eh	Command already in progress	
40h	Invalid Association ID	
41h	Invalid Connection ID	
FFh	Vendor specific (see bits 7:0)	
All others	Reserved	

reason code explanation: contains a value specified in table 17.

Table 17 – NVMe_LS reason code explanations

Value	Description	Applicable NVMe_LSs
00h	No additional explanation	
17h	Invalid OX_ID-RX_ID combination	
29h	Insufficient resources to support association or connection	
2Ah	Unable to supply requested data	
2Dh	Invalid payload length	
All others	Reserved	

Vendor specific: contains a vendor specific

8.2.4 Create Association Command descriptor

The format of the Create Association Command descriptor is specified in table 18.

Table 18 – Create Association Command descriptor

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	Descriptor tag = 0000 0003h			
1	Descriptor length			
2	NVMe_ERSP Ratio		Reserved	
3		Reserved		
11				
12	Controller ID (CNTLID)		Submission Queue Size (SQSIZE)	
13	Reserved			
14		Host Identifier (HOSTID)		
29				
30		Host NVMe Qualified Name (HOSTNQN)		
93				
94		NVM Subsystem NVMe Qualified Name (SUBNQN)		
157				
158		Reserved		
253				

Descriptor length: The Descriptor length field contains the length in bytes of the following payload.

NVMe_ERSP Ratio: contains the maximum number of completions over which at least one NVMe_ERSP shall be sent. The recommended value is ten percent of the Send Queue Size field value (e.g., send an NVMe_ERSP every NVMe_ERSP Ratio field value completions).

Controller ID: contains the controller identifier for the requested association as specified in the NVMe over Fabrics revision 1.0 Connect Command Capsule.

Submission Queue Size: contains the number of elements in the Admin Submission Queue to be created.

Host NVMe Qualified Name: contains the host NQN as specified in the NVMe over Fabrics revision 1.0 Connect Command Capsule.

NVM Subsystem NVMe Qualified Name: contains the NVM subsystem NQN as specified in the NVMe over Fabrics revision 1.0 Connect Command Capsule.

8.2.5 Create I/O Connection descriptor

The format of the Create I/O Connection descriptor is specified in table 19.

Table 19 – Create I/O Connection descriptor

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	Descriptor tag = 0000 0004h			
1	Descriptor length			
2	NVMe_ERSP Ratio		Reserved	
3	Reserved			
11				
12	Queue ID (QID)		Submission Queue Size (SQSIZE)	
13	Reserved			

Descriptor length: The Descriptor length field contains the length in bytes of the following payload.

Queue ID: contains the I/O queue identifier.

Submission Queue Size: contains the number of elements in the I/O Submission Queue to be created.

NVMe_ERSP Ratio: contains the maximum number of completions over which at least one NVMe_ERSP shall be sent. The recommended value is ten percent of the Send Queue Size field value (e.g., send an NVMe_ERSP every NVMe_ERSP Ratio field value completions).

8.2.6 Disconnect Command descriptor

The format of the Disconnect Command descriptor is specified in table 20.

Table 20 – Disconnect Command descriptor

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	Descriptor tag = 00000 0005h			
1	Descriptor length			
2	Reserved			Flags
3	Reserved			
4	MSB			
5		Identifier		LSB

Descriptor length: The Descriptor length field contains the length in bytes of the following payload.

Flags: The Flags field is specified in table 21.

Table 21 – Flags field

Bit	Description
7 to 1	Reserved
0	**Scope:** If the Scope bit is set to zero, then an association is to be terminated. If the Scope bit is set to one, then a connection is to be terminated/disconnected.

Identifier: If the Scope bit is set to zero, then the Identifier field is ignored. If the Scope bit is set to one, then the Identifier field contains an NVMe connection identifier to be terminated/disconnected.

8.2.7 Connection Identifier descriptor

The format of the Connection Identifier descriptor is specified in table 22.

Table 22 – Connection Identifier descriptor

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	Descriptor tag = 00000 0006h			
1	Descriptor length			
2	MSB			
3		Connection Identifier		LSB

Descriptor length: The Descriptor length field contains the length in bytes of the following payload.

Connection Identifier: contains a value that identifies the unique FC connection associating a host with a subsystem NQN.

8.2.8 Association Identifier descriptor

The format of the Association Identifier descriptor is specified in table 23.

Table 23 – Association Identifier descriptor

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	Descriptor tag = 00000 0007h			
1	Descriptor length			
2	MSB	Association Identifier		
3				LSB

Descriptor length: The Descriptor length field contains the length in bytes of the following payload.

Association Identifier: contains a value that identifies the host and controller association (see NVMe over Fabrics).

8.3 NVMe_LS reject (NVMe_RJT)

NVMe_RJT notifies the originator of an NVMe_LS request that the NVMe_LS request Sequence has been rejected. An NVMe_RJT may be a response Sequence to any NVMe_LS request.

Addressing: The D_ID field specifies the source of the NVMe_LS request being rejected. The S_ID field specifies the destination of the NVMe_LS request being rejected.

The format of the NVMe_RJT payload is specified in table 24.

Table 24 – NVMe reject payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	01h	00h	00h	00h
1	Descriptor list length			
2	MSB	Link Service Request Information descriptor		
3				
4				
5				LSB
6	MSB	Reject descriptor		
7				
8				
9				LSB

Descriptor list length: The Descriptor list length field contains the length in bytes of the following payload.

Link Service Request Information descriptor: contains a Link Service Request Information descriptor (see 8.2.2).

Reject descriptor: contains a Reject descriptor (see 8.2.3)

8.4 NVMe_LS accept (NVMe_ACC)

NVMe_ACC notifies the originator of an NVMe_LS request that the NVMe_LS request Sequence has been accepted. An NVMe_ACC may be a response Sequence to any NVMe_LS request.

Addressing: The D_ID field specifies the source of the NVMe_LS request being accepted. The S_ID field specifies the destination of the NVMe_LS request being accepted.

The format of the NVMe_ACC payload is specified in table 25.

Table 25 – NVMe accept payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	02h	00h	00h	00h
1	Descriptor list length			
2	MSB	Link Service Request Information descriptor		
3				
4				
5				LSB
6 to n	NVMe_LS descriptor(s)			

Descriptor list length: The Descriptor list length field contains the length in bytes of the following payload.

Link Service Request Information descriptor: contains a Link Service Request Information descriptor (see 8.2.2).

NVMe_LS descriptor(s): contains one or more NVMe_LS descriptors (see table 13).

8.5 Create Association

The format of the Create Association request payload is specified in table 26.

Table 26 – Create Association request payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	03h	Reserved	Reserved	Reserved
1	Descriptor list length			
2	MSB	Create Association Command descriptor		
25				LSB

Descriptor list length: The Descriptor list length field contains the length in bytes of the following payload.

Create Association Command descriptor: contains a Create Association Command descriptor (see 8.2.4).

The format of the Create Association accept payload is specified in table 27.

Table 27 – Create Association accept payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	02h	00h	00h	00h
1	Descriptor list length			
2	MSB	Link Service Request Information descriptor		
3				
4				
5				LSB
6	MSB	Association Identifier descriptor		
9				LSB
10	MSB	Connection Identifier descriptor		
13				LSB

Descriptor list length: The Descriptor length field contains the length in bytes of the following payload.

~~NVMe Link Service Request Information descriptor: contains an NVMe Link Service Request Information descriptor (see 8.2.2).~~

Link Service Request Information descriptor: contains a Link Service Request Information descriptor (see 8.2.2).

Association Identifier descriptor: contains an Association Identifier descriptor (see 8.2.8).

Connection Identifier descriptor: contains a Connection Identifier descriptor (see 8.2.7).

8.6 Create I/O Connection

The format of the Create I/O Connection request payload is specified in table 26.

Table 28 – Create I/O Connection request payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	04h	Reserved	Reserved	Reserved
1	Descriptor list length			
2	MSB	Association Identifier descriptor		
5				LSB
6	MSB	Create I/O Connection Command descriptor		
9				LSB

Descriptor list length: The Descriptor list length field contains the length in bytes of the following payload.

Association Identifier descriptor: contains an Association Identifier descriptor (see 8.2.8).

Create I/O Connection Command descriptor: contains a Create I/O Connection Command descriptor (see 8.2.5).

The format of the Create I/O Connection accept payload is specified in table 27.

Table 29 – Create I/O Connection accept payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	02h	00h	00h	00h
1	Descriptor list length			
2	MSB	Link Service Request Information descriptor		
3				
4				
5				LSB
10	MSB	Connection Identifier descriptor		
13				LSB

Descriptor list length: The Descriptor length field contains the length in bytes of the following payload.NVMe Link Service Request Information descriptor: contains an NVMe Link Service Request Information descriptor (see 8.2.2).

Link Service Request Information descriptor: contains a Link Service Request Information descriptor (see 8.2.2).

Connection Identifier descriptor: contains a Connection Identifier descriptor (see 8.2.7).

8.7 Disconnect

The Disconnect request is used to terminate a host and controller association or a connection.

The format of the Disconnect request payload is specified in table 30.

Table 30 – Disconnect request payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	05h	Reserved	Reserved	Reserved
1	Descriptor list length			
2	MSB	Association Identifier descriptor		
3				LSB
4	MSB	Disconnect Command descriptor		
9				LSB

Descriptor list length: The Descriptor list length field contains the length in bytes of the following payload.

Association Identifier descriptor: contains an Association Identifier descriptor (see 8.2.8).

Disconnect Command descriptor: contains a Disconnect Command descriptor (see 8.2.6).

The format of the Disconnect accept payload is specified in table 31.

Table 31 – Disconnect accept payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	02h	00h	00h	00h
1	Descriptor list length			
2	MSB	Link Service Request Information descriptor		
3				
4				
5				LSB

Descriptor list length: The Descriptor length field contains the length in bytes of the following payload.

Link Service Request Information descriptor: contains a Link Service Request Information descriptor (see 8.2.2).

9 NVMe over FC Information Unit (IU) usage and formats

9.1 Overview

Each NVMe over FC IU shall be contained in a single Sequence. Each Sequence carrying an NVMe IU shall contain only one IU.

NVMe over FC IUs and their characteristics are specified in table 32 for IUs sent to target NVMe_Ports.

Table 32 – NVMe over FC Information Units (IUs) sent to target NVMe_Ports

IU	Description	Data block		F/M/L	SI	M/O
		R_CTL field	Content			
T1	Command request	06h	NVMe_CMND	F	T	M
T2	Command request	06h	NVMe_CMND	F	H	O
T3	Data-Out action	01h	NVMe_DATA	M	T	M
T4	Confirm	03h	NVMe_CONF	L	T	O

Notes:
T2 is only permitted while NVMe_XFER_RDY IUs are disabled.
T2 allows optional Sequence streaming during write operations.
T4 is only permitted in response to an I4 or I6 frame (see table 33).

Key:

IU	Information Unit identifier
Content	Contents (payload) of data block
F/M/L	First/Middle/Last Sequence of Exchange (FC-FS-5)
	F First
	M Middle
	L Last
SI	Sequence Initiative: Held or Transferred (FC-FS-5)
	H Held
	T Transferred
M/O	Mandatory/Optional Sequence
	M Mandatory
	O Optional

NVMe over FC IUs and their characteristics are specified in table 33 for IUs sent to initiator NVMe_Ports.

Table 33 – NVMe over FC Information Units (IUs) sent to initiator NVMe_Ports

IU	Description	Data block		F/M/L	SI	M/O
		R_CTL field	Content			
I1	Data-Out delivery request	05h	NVMe_XFER_RDY (Write)	M	T	M
I2[b]	Data-In action	01h	NVMe_DATA	M	H	M
I3	Command response	07h	NVMe_RSP	L	T	M
I4[a]	Command response (NVMe_CONF IU request)	07h	NVMe_RSP	M	T	O
I5	Extended response	08h	NVMe_ERSP	L	T	M
I6[a]	Extended response (NVMe_CONF IU request)	08h	NVMe_ERSP	M	T	O

Notes:
a I4 or I6 is requested by not setting First/Middle/Last Sequence of Exchange (see FC-FS-5) to Last.
b I2 allows optional Sequence streaming to I2, I3, I4, I5, or I6.

Key:

IU	Information Unit identifier
Content	Contents (payload) of data block
F/M/L	First/Middle/Last Sequence of Exchange (FC-FS-5)
	F First
	M Middle
	L Last
SI	Sequence Initiative: Held or Transferred (FC-FS-5)
	H Held
	T Transferred
M/O	Mandatory/Optional Sequence
	M Mandatory
	O Optional

9.2 NVMe_CMND IU format

The NVMe_CMND IU contains an NVM command request and associated information. If an invalid combination of bits is set in the NVMe_CMND IU, then the target NVMe_Port shall respond with an NVMe_ERSP IU with the Status Code field set to INVALID FIELD (see table 41). The format of the NVMe_CMND IU is specified in table 34.

Table 34 – NVMe_CMND IU format

Bit / Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	SCSI ID (FDh)								FC ID (28h)								CMND IU Length															
1	Reserved																														Flags	
2	(MSB)								NVMe Connection Identifier																							
3																															(LSB)	
4	Command Sequence Number																															
5	Data Length																															
6	NVM Submission Queue Entry (64 bytes)																															
21																																
22	Reserved																															
23	Reserved																															

SCSI ID: The SCSI ID field shall be set to FDh (see SAM-6).

FC ID: The FC ID field shall be set to 28h to indicate FC-NVMe FC-4 Type.

CMND IU Length: The CMND IU Length field specifies the length in 4-byte words of the NVMe_CMND IU.

Flags: The Flags field bits are specified in table 35.

Table 35 – Flags field descriptors

Bit	Description
0	Write
1	Read
2 to 7	Reserved

If the Write bit is set to one, then the initiator NVMe_Port expects to transmit NVMe_DATA IUs to the target NVMe_Port (i.e., a write operation).

If the Read bit is set to one, then the initiator NVMe_Port expects to receive NVMe_DATA IUs from the target NVMe_Port (i.e., a read operation).

If the Read bit and Write bit are both set to zero, then there shall be no NVMe_DATA IUs and the Data Length field shall be set to zero.

A target NVMe_Port shall return an NVMe_ERSP IU with the Status Code field set to 01h (i.e., INVALID FIELD) if the following protocol errors are detected:

a) a read operation has the Read bit set to zero or the Write bit set to one;
b) a write operation has the Write bit set to zero or the Read bit set to one;
c) the Read bit and Write bit are both set to one; or
d) the Read bit and Write bit are both set to zero and the Data Length field value is not zero.

NVMe Connection Identifier: The NVMe Connection Identifier field identifies the unique FC session associating a host with a subsystem NQ.

Command Sequence Number (CSN): The Command Sequence Number field enables the receiving NVMe_Port to maintain proper command ordering as specified in 4.7.2.

Data Length: The Data Length field indicates the length of the data to be read or written for the command.

For a read command, the Data Length field contains a count of the maximum number of all bytes to be transferred from the target NVMe_Port in NVMe_DATA IU payloads by the command.

For a write command, the Data Length field contains a count of the maximum number of all bytes to be transferred to the Target NVMe_Port in NVMe_DATA IU payloads by the command.

A Data Length field value of zero indicates that no data transfer is expected regardless of the state of the ~~R and W~~ bits and that no NVMe_XFER_RDY or NVMe_DATA IUs shall be transferred.

NVM Submission Queue Entry: The NVM Submission Queue Entry field contains an NVM Submission Queue Entry as specified in NVMe over Fabrics revision 1.0.

9.3 NVMe_XFER_RDY IU format

The format of the NVMe_XFER_RDY IU is specified in table 36.

Table 36 – NVMe_XFER_RDY IU format

Bit / Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Relative Offset																															
1	Burst Length																															
2	Reserved																															

Relative Offset: The Relative Offset field contains a value specifying the relative offset in the Parameter field for the first data byte of the requested NVMe_DATA IU.

The Relative Offset field shall have a value that is a multiple of 4 (i.e., each NVMe_DATA IU shall begin on a word boundary).

Burst Length: The Burst Length field contains a value indicating the amount of buffer space prepared for all bytes to be transferred in the next NVMe_DATA IU and requests the transfer of an NVMe_DATA IU of that length from the initiator NVMe_Port.

The sum of the value of the Burst Length field and the value of the Relative Offset field shall not exceed the value of the Data Length field. The value in the Burst Length field shall not be zero.

9.4 NVMe_DATA IU format

9.4.1 NVMe_DATA IU overview

The data associated with a particular NVMe over FC I/O operation is transmitted in the same Exchange that sent the NVMe_CMND IU requesting the transfer.

NVMe over FC data transfers may be performed by one or more data delivery requests with the following constraints:

a) If the First Burst Supported bit is set to one, the first burst NVMe_DATA IU shall be no longer than the First Burst Size field value (see 6.3.4);
b) NVMe_DATA IUs for write data, excluding the first burst NVMe_DATA IU, shall be the length specified in the Burst Length field value in the corresponding NVMe_XFER_RDY IU that was received; and
c) NVMe_DATA IUs for read data shall be no longer than the Data Length field in the received NVMe_CMND IU.

If more than one NVMe_DATA IU is used to transfer the data, the relative offset value in the Parameter field is used to ensure that the NVM data is reassembled in the proper order.

If the NVMe_DATA IU is for first burst write data, then the relative offset for the NVMe_DATA IU shall be set to zero. If the first frame transmitted of the first burst NVMe_DATA IU has a relative offset that is not zero, then the target NVMe_Port shall return an NVMe_ERSP IU with the Status Field of the NVMe CQE set to TRANSPORT ERROR (see table 41).

If an NVMe_XFER_RDY IU is used to describe a data transfer and the first frame transmitted of the requested NVMe_DATA IU has a relative offset that differs from the value in the Relative Offset field of the NVMe_XFER_RDY IU, then the target NVMe_Port shall return an NVMe_ERSP IU with the Status Field of the NVMe CQE set to TRANSPORT ERROR (see table 41).

All write data NVMe_DATA IUs excluding the first burst NVMe_DATA IU if applicable, shall be preceded by an NVMe_XFER_RDY IU containing a standard data descriptor payload that indicates the location and length of the data delivery. If the First Burst Supported bit is set to one in the PRLI NVMe over FC Service Parameter page request and accept (see 6.3), then the first NVMe_DATA IU may be transmitted without a preceding NVMe_XFER_RDY IU.

If more than one read data NVMe_DATA IU is used to transfer the data, the relative offset of the NVMe_DATA IU may be specified in any order (i.e., there is no requirement that successive read data NVMe_DATA IUs specify increasing and successive relative offsets).

If more than one NVME_XFER_RDY is used to transfer write data, the relative offset of the NVMe_XFER_RDY may be specified in any order (i.e., there is no requirement that successive NVMe_XFER_RDY IUs specify increasing and successive relative offsets). Data overlay is not allowed except to retransmit first burst write data.

If error conditions occur that prevent the transfer of data in the middle of a NVMe_DATA IU, then the target NVMe_Port NVMe_ERSP IU Transferred Data Length field (see table 38) shall indicate a value that reflects the amount of data transferred up until the point of the error, and the target NVMe_Port shall set the NVMe CQE Status to an appropriate value to reflect the error.

9.4.2 NVMe_DATA IUs for read and write operations

During any data transfer, the initiator NVMe_Port shall have available a buffer of the length specified by the Data Length field in the NVMe_CMND IU. The buffer contains data to be transferred to the target NVMe_Port if the operation is a write operation (i.e., an operation that uses the Data-Out action, IU T). The buffer receives the data if the operation is a read operation (i.e., an operation that uses the Data-In action, IU I). The target NVMe_Port shall not request or deliver data outside the buffer length defined by the Data Length field value.

If the command requested that data beyond the length specified by the Data Length field be transferred, then the target NVMe_Port shall:

a) transfer no data and return NVMe_ERSP IU with the Transferred Data Length set to zero and the Status Field of the NVMe CQE set to 04h (i.e., Data Transfer Error); or
b) may transfer data and return NVMe_ERSP IU with the Transferred Data Length set to amount of data transferred and Status Field of the NVMe CQE set to 04h (i.e., Data Transfer Error).

During a write operation that is sending first burst data, the initiator NVMe_Port indicates that it has transferred all the first burst data by transferring Sequence Initiative to the target NVMe_Port.

The initiator NVMe_Port shall not transfer data outside the buffer length defined by the Data Length field value. If the initiator NVMe_Port transfers an amount of first burst data that exceeds the Data Length in the NVMe_CMND IU then the target NVMe_Port shall discard the excess bytes. The target NVMe_Port shall then return an NVMe_ERSP IU with the Transferred Data Length set to the number of bytes received and not discarded, and the Status Field of the NVMe CQE set to 04h (i.e., Data Transfer Error).

Upon completion of all data transfer for the command as determined by the target NVMe_Port the Transferred Data Length field value in the NVMe_ERSP IU shall be set to the number of bytes transferred, as adjusted by first burst retransmission, if applicable.

9.4.3 NVMe_Port transfer byte counting

The initiator NVMe_Port shall:

a) maintain a byte count of transferred data for the command;
b) ~~the byte count shall be set to zero upon~~ transmitting the NVMe_CMND IU;
c) ~~when receiving read data the byte count shall be incremented~~ by the amount of payload in each successfully received NVMe_DATA IU;
d) ~~when transferring write data the byte count shall be incremented by~~ the amount of payload in each successfully transmitted NVMe_DATA IU;
e) ~~if first burst was transmitted for the command and a NVMe_XFER_RDY IU is received with relative offset set to zero, then byte count shall be decremented by the amount of first burst data transmitted;~~ and
f) ~~if an NVMe_ERSP IU is received with a Transferred Data Length field value that does match its transferred byte count and the NVMe CQE Status Field set to zero (i.e., Successful Completion) or the initiator NVMe_Port receives an NVMe_RSP IU and its transferred byte count does not match the Data Length field value in the NVMe_CMND IU, the initiator NVMe_Port shall interact with the NVMe implementation to reflect a failure of the NVMe queue for the NVMe_CMND IU~~

The target NVMe_Port shall:

a) maintain a byte count of transferred data for the command;

b) the byte count shall be set to zero upon receiving the NVMe_CMND IU;
c) when transmitting read data the byte count shall be incremented by the amount of payload in each successfully transmitted NVMe_DATA IU;
d) when receiving write data the byte count shall be incremented by the amount of payload in each successfully received NVMe_DATA IU;
e) if first burst was received and discarded the byte count shall remain zero;
f) an NVMe_RSP IU shall only be sent if the byte count is equal to the data length value specified in the NVMe_CMND IU; and
g) when sending an NVMe_ERSP IU the Transferred Data Length field shall be set to the byte count.

9.4 NVMe_DATA IU use of fill bytes

During transfer of data in response to an NVMe_CMND_IU with the Read bit set to one and the Write bit set to zero, all frames of NVMe_DATA_IUs except the frame with the highest relative offset within the Data-In Buffer shall have no fill bytes.

During transfer of data in response to an NVMe_CMND_IU with the Write bit set to one and the Read bit set to zero, all frames of NVMe_DATA_IUs except the frame with the highest relative offset within the Data-Out Buffer shall have no fill bytes.

9.5 NVMe_RSP IU format

The format of the NVMe_RSP IU is specified in table 37. ~~The NVMe_RSP IU may be used for Completion Queue Entries containing Status Code Type field set to 0h (i.e., Generic Command Status) (see NVM Express revision 1.2.1) and Status Code field set to 00h (i.e., Successful Completion) (see NVM Express revision 1.2.1).~~

Table 37 – NVMe_RSP IU format

Bit / Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	00h								00h								00h								00h							
1	00h								00h								00h								00h							
2	00h								00h								00h								00h							

9.6 NVMe_ERSP IU format

The format of the NVMe_ERSP IU is specified in table 37. ~~The NVMe_ERSP IU shall be used for all Completion Queue Entries containing Status Code Type field not set to 0h (i.e., Generic Command~~

Status) (see NVM Express revision 1.2.1) and Status Code field not set to zero (i.e., Successful Completion), and may be used to periodically provide a SQ Head Pointer value.

Table 38 – NVMe_ERSP IU format

Bit / Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Status Code								Reserved								ERSP IU Length															
1	Response Sequence Number																															
2	Transferred Data Length																															
3	Reserved																															
4	NVM Completion Queue Entry (16 bytes)																															
5																																
6																																
7																																
8	NVM Response Additional Data (if any, n words)																															
n+8																																

Status Code: the Status Code field contains an NVMe over FC specific status value specified in table 39.

Table 39 – Status Code field values

Value	Name	Description
00h	SUCCESS	No status.
01h	INVALID FIELD	NVMe_CMND IU field is invalid.
02h	INVALID CONNECTION ID	Connection Identifier is invalid.
Others	-	Reserved

ERSP IU Length: The ERSP IU Length field specifies the length in 4-byte words of the NVMe_ERSP IU.

Response Sequence Number: The Response Sequence Number field enables the receiving NVMe_Port to maintain proper response ordering as specified in 4.7.3.

Transferred Data Length: Specifies the total number of bytes transferred in Data (read) or Data (write) IUs on behalf of the command. This field shall be set to zero if no data has been transferred for the command.

NVM Completion Queue Entry: The NVM Completion Queue Entry field contains an NVM Completion Queue Entry as specified in NVMe over Fabrics revision 1.0.

9.7 NVMe_CONF IU format

The NVMe_CONF IU has no payload. It is used as specified in 4.10 for an initiator NVMe_Port to confirm the receipt of an NVMe_RSP IU or NVMe_ERSP IU from a target NVMe_Port. The frame

shall be transmitted by an initiator NVMe_Port if the confirmed completion protocol is supported by both the target NVMe_Port and the initiator NVMe_Port and the confirmation has been requested by the target NVMe_Port.

10 NVMe over Fabrics

10.1 Discovery

10.1.1 Overview

A target NVMe_Port that participates in FC-NVMe Fabric discovery shall support an NVMe Discovery Service (see NVMe over Fabrics revision 1.0) that reports NVM subsystems local to the target NVMe_Port.

NOTE 3 – In a Fabric topology, discovery of NVM subsystems behind target NVMe_Ports that do not support an NVMe Discovery Service is outside the scope of this standard.

10.1.2 Discovery Log Page Entry

The Discovery Log Page for NVMe over FC is specified in table 40.

Table 40 – Discovery Log Page for NVMe over FC

Bit / Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Transport Type (TRTYPE)								Address Family (ADRFAM)								Subsystem Type (SUBTYPE)								Transport Requirements (TREQ)							
1	Port ID (PORTID)																Controller ID (CNTLID)															
2	Admin Max SQ Size (ASQSZ)																Reserved															
3 to 7	Reserved																															
8	Transport Service ID (TRSVCID) (32 bytes)																															
15																																
16 to 63	Reserved																															
64	NVMe Qualified Name (SUBNQN) (256 bytes)																															
127																																
128	Transport Address (TRADR) (256 bytes)																															
191																																
192	Transport Specific Address Subtype (TSAS) (256 bytes)																															
255																																

Transport Type: shall be set to 02h (see NVMe over Fabrics revision 1.0).

Address Family: shall be set to 04h (see NVMe over Fabrics revision 1.0).

Subsystem Type: shall be set to 01h (i.e., referral) or 02h (i.e., NVM subsystem) (see NVMe over Fabrics revision 1.0).

Transport Requirements: see NVMe over Fabrics revision 1.0.

Port ID: shall be set to an NVM subsystem specific value (see NVMe over Fabrics revision 1.0).

Controller ID: shall be set to an NVM subsystem specific value (see NVMe over Fabrics revision 1.0).

Admin Max SQ Size: shall be set to an NVM subsystem specific value (see NVMe over Fabrics revision 1.0).

Transport Service ID: shall be set to "none" (see NVMe over Fabrics revision 1.0).

NVMe Qualified Name: shall be set to the subsystem NQN (see NVMe over Fabrics revision 1.0).

Transport Address: shall be set to "nn-WWNN:pn-WWPN" where:

a) WWNN is the World Wide Node Name of the target NVMe_Port; and
b) WWPN is the World Wide Port Name of the target NVMe_Port.

The WWNN and WWPN are the VPD SII values of the Worldwide_Name(s).

Transport Specific Address Subtype: shall be set to all zeroes.

2 Transport specific status

The value range B0-BFh are defined by NVMe Express (see NVMe over Fabrics revision 1.0) for transport specific errors.

Transport specific status values for NVMe over FC are specified in table 41.

Table 41 – NVMe over FC transport specific status values

Value	Name	Description
B0h	TRANSPORT ERROR	Generic failure
B1h	TRANSPORT ABORTED	I/O failure due to ABTS-LS
B2h to BFh	Reserved	

11 Link error detection and error recovery procedures

11.1 Overview

This standard provides several mechanisms for NVMe over FC devices to identify protocol errors caused by frames and responses that have been corrupted and discarded in accordance with the requirements of FC-FS-5. See 11.2 for a list of these mechanisms.

11.2 Error detection

An initiator NVMe_Port shall detect the following:

a) a Sequence error (see FC-FS-5) in a Sequence transmitted from a target NVMe_Port to an initiator NVMe_Port;
b) an NVMe_XFER_RDY IU is received on an Exchange where the NVMe_CMND IU flags had the Read bit set to one.
c) a command is completed with an NVMe_ERSP IU and the initiator data transfer byte count value is not equal to the NVMe_ERSP IU Transferred Data Length field value; and
d) a command is completed with an NVMe_RSP IU and the initiator data transfer byte count value is not equal to the NVMe_CMND IU Data Length field value.

A target NVMe_Port shall detect the following:

a) a Sequence error (see FC-FS-5) in a Sequence transmitted from an initiator NVMe_Port to a target NVMe_Port; and
b) an NVMe_DATA IU is received with a starting Relative Offset value that is not set to the same Relative Offset value contained in the last NVMe_XFER_RDY IU transmitted to the initiator.

Upon detection of an error, the detecting NVMe_Port may transmit an ABTS-LS to terminate the failing Exchange and recover the associated Exchange resources (see 11.3). If an NVMe I/O Exchange is terminated by an ABTS-LS, as it potentially causes loss of a SQE, CQE or data for an NVMe command. Use of ABTS-LS shall cause the termination of the NVMe over FC connection and NVMe over FC association that were associated with NVME I/O Exchang

11.3 Exchange level recovery using ABTS-LS

11.3.1 ABTS-LS overview

ABTS-LS is an FC-FS-5 protocol that recovers NVMe_Port resources associated with an Exchange that is being terminated because of an error. If an NVMe I/O Exchange is terminated by an ABTS-LS, as it potentially causes loss of a SQE, CQE or data for an NVMe command shall cause the termination of the NVMe over FC connection and NVMe over FC association that are associated with NVMe I/O Exchange. Refer to the actions specified in clause 4 for termination of FC-NVMe transport connections and associations.

All NVMe over FC compliant initiator and target NVMe_Ports shall be capable of transmitting an Abort Exchange (i.e., ABTS-LS), and capable of accepting and processing an ABTS-LS.

11.3.2 Initiating NVMe_Port Exchange termination

The NVMe_Port terminating the Exchange transmits an ABTS-LS to the D_ID of the corresponding NVMe_Port of the Exchange being terminated. The ABTS-LS shall be generated using the OX_ID field and RX_ID field values of the Exchange to be aborted. FC-FS-5 allows ABTS-LS to be transmitted by an NVMe_Port regardless of whether or not it has Sequence Initiative. Following the

transmission of ABTS-LS, any Device_Data Frames received for the Exchange being terminated shall be discarded until the BA_ACC with the F_CTL field Last_Sequence bit set to one (i.e., last Sequence of the Exchange) is received from the corresponding NVMe_Port.

Exchange termination may not take effect immediately (e.g., if ABTS-LS is sent following transmission of a read command, an NVMe_Port may receive some or all of the requested read data before receiving the BA_ACC for the ABTS-LS). The NVMe_Port shall be capable of receiving this data and providing BB_Credit in order for the corresponding NVMe_Port to transmit the BA_ACC.

If a BA_ACC, BA_RJT, LOGO ELS, or PRLO ELS is not received from the corresponding NVMe_Port within two times R_A_TOV, then second level error recovery (see 11.4) shall be performed.

11.3.3 Recipient NVMe_Port response to Exchange termination

If an ABTS-LS is received by an NVMe_Port, it shall abort the designated Exchange and return one of the following responses:

a) the receiving NVMe_Port shall discard the ABTS-LS and transmit a LOGO ELS if the Nx_Port issuing the ABTS-LS is not currently logged in (i.e., no N_Port Login exists);
b) the receiving NVMe_Port shall return BA_RJT with the F_CTL field Last_Sequence bit set to one (i.e., last Sequence of the Exchange) if the received ABTS-LS contains an assigned RX_ID field value and a FQXID that is unknown to the receiving NVMe_Port; or
c) the receiving NVMe_Port shall return BA_ACC with the F_CTL field Last_Sequence bit set to one (i.e., last Sequence of the Exchange).

Upon transmission of any of the above responses, the receiving NVMe_Port may reclaim any resources associated with the designated Exchange ~~after R_A_TOV has elapsed~~.

If the RX_ID field is set to FFFFh, then receiving NVMe_Ports shall qualify the FQXID of the ABTS-LS based only upon the combined values of the D_ID field, S_ID field, and the OX_ID field, not the RX_ID field.

11.3.4 Error recovery

NVMe over Fabrics revision 1.0 specifies unrecoverable transport errors should result in termination of the NVMe transport connection, and association between the host and controller.

As such, an initiator NVMe_Port or target NVMe_Port, after detecting an error in an Exchange, shall stop all processing of the Exchange and its associated NVMe command, and the detecting NVMe_Port shall issue an ABTS-LS for the Exchange (see 11.3).

11.3.5 Additional error recovery by initiator NVMe_Port

If an initiator NVMe_Port detects a Sequence error, it shall discard the Sequence(s) based on the Exchange error policy specified by the F_CTL field Abort Sequence Condition (see FC-FS-5) bits in the first frame of the Exchange.

The initiator NVMe_Port shall defer to upper level protocol mechanisms to determine lack of continued response by the target NVMe_Port for a particular Exchange and the error recovery actions that are to be taken in such situations. For example, the NVMe protocol layer may maintain an “io completion timer”, that upon expiration, proceeds to send a NVMe Admin ABORT command to terminate the corresponding NVMe command. The NVMe protocol layer may also detect a lack of completion for a command and revert to resets of the NVMe controller, which will terminate the FC-

NVMe association, its FC-NVMe connections, as well as all outstanding I/O operations on those connections. The I/O operations, which correspond to FC-NVMe Exchanges, shall be terminated by the initiator NVMe_Port by an ABTS-LS.

11.3.6 Additional error recovery by target NVMe_Port

If a target NVMe_Port detects a Sequence error, it shall discard the Sequence(s) based on the Exchange error policy specified by the F_CTL field Abort Sequence Condition (see FC-FS-5) bits in the first frame of the Exchange.

The target NVMe_Port shall not attempt recovery for Sequence errors. The target NVMe_Port shall depend on the initiator NVMe_Port for recovery.

Target NVMe_Ports shall implement IR_TOV (see 12.3) to facilitate recovery of resources allocated to an initiator NVMe_Port that is no longer responding. The target NVMe_Port may transmit a LOGO ELS to the initiator NVMe_Port and terminate all NVMe over FC associations, all NVMe over FC connections, and all open Exchanges for that initiator NVMe_Port upon IR_TOV timeout without the initiator NVMe_Port transmitting any expected Sequence for any open Exchange at this target NVMe_Port (e.g., NVMe over FC write Data-In response to an NVMe_XFER_RDY IU).

11.4 Second-level error recovery

11.4.1 ABTS error recovery

If a response to an ABTS is not received within two times R_A_TOV, then the NVMe_Port may transmit the ABTS again, attempt other retry operations allowed by FC-FS-5, or explicitly logout the corresponding NVMe_Port. If those retry operations attempted are unsuccessful, then the NVMe_Port shall explicitly logout (i.e., transmit a LOGO ELS) the corresponding NVMe_Port. All outstanding Exchanges as well as all NVMe over FC connections and NVMe over FC associations, with the corresponding NVMe_Port are terminated at the NVMe_Port.

11.5 Responses to frames before port login or process login

If a target NVMe_Port receives an NVMe FC-4 Link Service or NVMe_CMND IU from an NVMe_Port that is not successfully logged into the target NVMe_Port using either an implicit or explicit login, then it shall discard the Link Service or NVMe_CMND IU and, in a new Exchange, transmit a LOGO ELS request to that NVMe_Port. No Exchange is created in the target NVMe_Port for the discarded request, and the Originator of the discarded request terminates the Exchange associated with the discarded request and any other open Exchanges for the target NVMe_Port transmitting the LOGO ELS.

If a target NVMe_Port receives an NVMe FC-4 Link Service or NVMe_CMND IU from an NVMe_Port that has not successfully completed either implicit or explicit Process Login with the target NVMe_Port, then it shall discard the Link Service or NVMe_CMND IU and transmit a PRLO ELS to the initiator NVMe_Port. No Exchange is created in the recipient NVMe_Port for the discarded request, and the Originator of the discarded request terminates the Exchange associated with the discarded request.

If an NVMe over FC device receives a frame of category 0001b or 0011b (i.e., solicited data or solicited control) and the NVMe over FC device has not performed successful implicit or explicit login and Process Login with the source of the frame, then the NVMe over FC device shall discard and ignore the content of the frame. If login is not completed, then the NVMe over FC device may transmit a LOGO ELS request to the source of the unexpected frame. If login is completed, but Process Login

is not completed, then the NVMe over FC device may transmit a PRLO ELS request to the source of the unexpected frame.

12 Timers for operation and recovery

12.1 Overview

This clause indicates the use of timers defined by other standards in performing the NVMe recovery procedures. In addition, the clause defines those timers used only by this standard.

Table 42 – Timers summary

Timer	Implementation		Description	Default Value	Ref
	Initiator NVMe_Port	Target NVMe_Port			
R_A_TOV	M	M	Resource_Allocation_Timeout Value	see FC-FS-5	12.2
IR_TOV	n/a	M	Initiator Response Timeout Value	2 s[a]	12.3

Keywords:
M - Manadatory
O - Optional

a) This value is not configurable.

12.2 Resource Allocation Timeout Value (R_A_TOV)

~~R_A_TOV is the minimum amount of time that a Sequence Initiator shall wait before reusing the Sequence_Qualifier associated with an aborted Sequence. The Sequence_Qualifier is composed of the S_ID field, D_ID field, OX_ID field, RX_ID field, and SEQ_ID field.~~

An NVMe_Port shall not wait R_A_TOV after receiving a BA_ACC to an ABTS-LS before reusing the Sequence_Qualifier.

The Originator of an NVMe FC-4 Link Service Exchange shall detect an Exchange error following Sequence Initiative transfer if the reply Sequence is not received within a timeout interval equal to twice the value of R_A_TOV.

12.3 Initiator Response Timeout Value (IR_TOV)

IR_TOV is the minimum time a target NVMe_Port shall wait for an initiator NVMe_Port response following transfer of Sequence Initiative from the target NVMe_Port to the initiator NVMe_Port (e.g., following transmission of the NVMe_XFER_RDY IU during a write command). If the initiator NVMe_Port does not send a response within IR_TOV of the transfer of Sequence Initiative, then a target NVMe_Port may send an ABTS-LS to terminate the Exchange.

Annex A
(informative)
NVMe Information Unit examples

A.1 Overview

The byte order of Fibre Channel standards is big-endian. As such, multi-byte values are transmitted or stored in host memory with the Most Significant Byte (MSB) first. For example, when transmitting a four byte word, the Most Significant Byte (i.e., corresponding to bits 31:24) is placed first, followed by the next lesser significant byte (i.e., corresponding to bits 24:16), followed by the next lesser significant byte (i.e., corresponding to bits 15:8), followed by the Least Significant Byte (i.e., corresponding to bits 7:0).

In contrast, the byte order of the NVM Express and NVM Express over Fabrics specifications is little-endian. As such, multi-byte values are transmitted or stored in host memory with the Least Significant Byte (LSB) first. For example, when transmitting a four byte word, the Least Significant Byte (i.e., corresponding to bits 7:0) is placed first, followed by the next more significant byte (i.e., corresponding to bits 15:8), followed by the next more significant byte (i.e., corresponding to bits 23:16), followed by the Most Significant Byte (i.e., corresponding to bits 31:24).

In the FC-NVME standard, the NVMe_CMND IU and NVMe_ERSP IU are defined with Fibre Channel specific areas which then encapsulate the NVM Express specific area. When transmitting or storing the IU payload, the IU will be treated as a raw bytestream and each area will be stored in its native endianness. The Fibre Channel area is stored in big-endian and the NVM Express area is stored in little-endian.

To further clarify, the following diagrams document the IU content with the NVM Express areas explicitly enumerated and converted to diagrams that are consistent with the Fibre Channel standard and viewed as big-endian in their entirety.

A.2 NVMe_CMND IU payload

Table A.1 illustrates a NVMe_CMND IU with the NVM Express SQE area explicitly converted to its representation in a payload that is big-endian in nature. The SQE follows the format for a NVM Command Set as defined in section 4.2 of the NVMe Express 1.2b specification. The SGL1 field,

contained in words 12-15, illustrates a SGL Data Block Descriptor. As a reminder, all multi-byte fields for the NVM Express area are stored LSB first (i.e., leftmost) proceeding to MSB last (i.e., rightmost).

Table A.1 - NVMe_CMND IU with NVM Express SQE format

Bit / Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	SCSI ID (FDh)								FC ID (28h)								CMND IU Length															
1	Reserved																														Flags	
2	(MSB)								NVMe Connection Identifier																							
3																															(LSB)	
4	Command Sequence Number																															
5	Data Length																															
6	Opcode (OPC)								FUSE			Reserved			PSDT		(LSB)					Connection ID (CID)										(MSB)
7	(LSB)								NSID																							(MSB)
8	Reserved																															
9	Reserved																															
10	(LSB)								MPTR																							
11																																(MSB)
12	(LSB)								Address																							
13																																(MSB)
14	(LSB)								Length																							(MSB)
15	Reserved																								Zero						SGL Descriptor Type	
16	Command Dword 10																															
17	Command Dword 11																															
18	Command Dword 12																															
19	Command Dword 13																															
20	Command Dword 14																															
21	Command Dword 15																															
22	Reserved																															
23	Reserved																															

A.3 NVMe_ERSP IU payload

Table A.1 illustrates a NVMe_ERSP IU with the NVM Express CQE area explicitly converted to its representation in a payload that is big-endian in nature. There is no NVM Express Response Additional Data. The CQE follows the format for a Completion Queue Entry as specified in section 4.6

of the NVMe Express 1.2b specification. As a reminder, all multi-byte fields for the NVM Express area are stored LSB first (i.e., leftmost) proceeding to MSB last (i.e., rightmost).

Table A.2 - NVMe_ERSP IU with NVM Express CQE format

Bit / Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Status Code								Reserved								ERSP IU Length															
1	Response Sequence Number																															
2	Transferred Data Length																															
3	Reserved																															
4	Command Specific																															
5	Reserved																															
6	(LSB)	SQ Head Pointer						(MSB)									(LSB)	SQ Identifier														(MSB)
7	(LSB)	Command Identifier						(MSB)									P	(lsb)	Status Field													(msb)

Annex B
(informative)
NVMe over FC command IU examples

B.1 Overview

The steps given in the following examples indicate the normal exchange of NVMe over FC IUs corresponding to the handling of an NVMe command. The examples are not all inclusive. There may be additional transmissions to detect and recover from FC frame loss or error, or to communicate command response reception.

B.1.1 NVMe command with no payload

The following procedure illustrates the basic steps for an NVMe command which does not have payload. The command is initiated by an SQE and completed by a CQE.

1) The initiator NVMe_Port allocates an Exchange and transmits a NVMe_CMND IU. The SQE within the IU contains the command. The IU indicates the association and connection of the command;
2) The target NVMe_Port receives the IU and interacts with the NVMe layer to initiate processing of the command;
3) The NVMe layer finishes command processing, constructs a corresponding CQE, and posts it to the FC-NVMe layer; and
4) The target NVMe_Port transmits an NVMe_RSP IU or NVMe_ERSP IU to communicate the CQE contents relative to the Exchange/NVMe command back to the initiator NVMe_Port. The Exchange is completed.

B.1.2 NVMe command with read payload

The following procedure illustrates the basic steps for an operation where command data is passed from the target NVMe_Port to the initiator NVMe_Port (i.e., read operation). The command is initiated by an SQE and completed by a CQE. Data transfer for the command is initiated by the target NVMe_Port. The read data may be response data, on operations that are not LBA-relative, or LBA read data.

1) The initiator NVMe_Port allocates an Exchange and transmits an NVMe_CMND IU. The SQE within the IU contains the command. The IU indicates the association and connection of the command;
2) The target NVMe_Port receives the IU and interacts with the NVMe layer to initiate processing of the command;
3) The NVMe layer makes one or more requests to transfer the read data to the initiator. For each request, the target NVMe_Port transmits an NVMe_DATA IU containing the provided read data;
4) The NVMe layer finishes command processing, constructs a corresponding CQE, and posts it to the FC-NVMe layer; and
5) The target NVMe_Port transmits an NVMe_RSP_IU or NVMe_ERSP_IU to communicate the CQE contents relative to the Exchange/NVMe command back to the initiator NVMe_Port. The Exchange is completed.

B.1.3 NVMe write command with no first burst

The following procedure illustrates the basic steps for an operation where command data is passed from the initiator NVMe_Port to the target NVMe_Port (i.e., write operation). The command is initiated by an SQE and completed by a CQE. Data transfer for the command is initiated by the target NVMe_Port. The write data may be command data, on operations that are not LBA-relative, or LBA write data.

1) The initiator NVMe_Port allocates an Exchange and transmits a NVMe_CMND IU. The SQE within the IU contains the command. The IU indicates the association and connection of the command;
2) The target NVMe_Port software receives the IU and interacts with the NVMe software to initiate processing of the command;
3) The NVMe layer makes one or more requests to transfer the write data from the initiator NVMe_Port. For each request:
   a) the target NVMe_Port transmits an NVMe_XFER_RDY IU indicating the desired data range; and
   b) the initiator NVMe_Port transmits an NVMe_DATA IU containing the requested write data;
4) The NVMe layer finishes command processing, constructs a corresponding CQE, and posts it to the FC-NVMe layer; and
5) The target NVMe_Port transmits an NVMe_RSP_IU or NVMe_ERSP_IU to communicate the CQE contents relative to the Exchange/NVMe command back to the initiator NVMe_Port. The Exchange is completed.

B.1.4 NVMe write command with first burst

The following procedure illustrates the basic steps for an operation where command data is passed from the initiator NVMe_Port to the target NVMe_Port (i.e., write operation). In this example, an initial burst of data is transmitted to the target NVMe_Port along with the command. The command is initiated by an SQE and completed by a CQE. Data transfer for the command, except for the initial burst, is initiated by the target NVMe_Port. The write data may be command data, on operations that are not LBA-relative, or LBA Write data.

1) The initiator NVMe_Port allocates an Exchange and transmits an NVMe_CMND IU. The NVMe_CMND IU does not pass Sequence Initiative. The SQE within the IU contains the command. The IU indicates the association and connection of the command;
2) The initiator NVMe_Port transmits an NVMe_DATA IU containing an initial burst of write data. The write data starts at offset 0. The length of the data is subject to the values negotiated by the PRLI ELS;
3) The target NVMe_Port software receives the NVMe_CMND IU and interacts with the NVMe layer to initiate processing of the command;
4) The target NVMe_Port receives the NVMe_DATA IU and interacts with the NVMe layer for handling;
5) If additional write data is to transferred, the NVMe layer makes one or more requests to transfer the write data from the initiator NVMe_Port. For each request:
   a) the target NVMe_Port transmits an NVMe_XFER_RDY IU indicating the desired data range; and
   b) the initiator NVMe_Port transmits an NVMe_DATA IU containing the requested write data;

6) The NVMe layer finishes command processing, constructs a corresponding CQE, and posts it to the FC-NVMe layer; and
7) The target NVMe_Port transmits an NVMe_RSP_IU or NVMe_ERSP_IU to communicate the CQE contents relative to the Exchange/NVMe command back to the initiator. The Exchange is completed.

Annex C
(informative)
NVMe over FC initialization and device discovery

C.1 NVMe over FC device discovery procedure

C.1.1 Initiator discovery of switched Fabric-attached target NVMe_Ports

The following procedure may be used by initiator NVMe_Ports for discovering NVMe over FC devices in a switched Fabric topology.

Depending on the specific configuration and the management requirements, any step other than steps 1 through 3 may be omitted and may be performed using actions outside this standard or the referenced standards.

1) Perform Fabric Login;
2) Login with the Name Server;
3) Register information with Name Server:
   a) FC-4 TYPEs object (see 7.2); and
   b) FC-4 Features object (see 7.3).
4) Register for State Change Notification with the Fabric Controller (see FC-LS-3);
5) Issue a GID_FF (see FC-GS-8) query to the Name Server with the Domain_ID Scope and Area_ID Scope fields set to zero, the FC-4 Feature Bits field set to 04h (i.e., Discovery Service supported), and the Type code field set to 28h (i.e., NVMe over FC). This query obtains a list of the Port Identifiers (see FC-GS-8) of devices that support the NVMe over FC protocol, and a Discovery Service (see NVMe over Fabrics);
6) For each Port Identifier returned in the accept CT_IU for the GID_FF which returned all N_Port Id's with support for Type 0x28 and FC-4 Feature Bits 04h (i.e., NVMe Discovery Service supported):
   i) the NVMe layer initiates a session with the NVMe Discovery Service:
      1) the initiator NVMe_Port ensures there is a login with the FC target NVMe_Port. Note: if there is already an active login between the initiator NVMe_Port and target NVMe_Port's, these steps may be skipped:
         i) send PLOGI;
         ii) send PRLI with Type field set to 28h;
      2) FC-NVMe layer creates an association and the initial Admin Queue connection:
         i) send Create Association NVMe_LS to the Discovery Service subsystem.
      3) the NVMe layer issues a NVMe Connect command via the newly created transport Admin Queue connection. The Connect command is to create the Admin Queue.
         i) the Connect command is classified as a write operation (see B.1.3 or B.1.4).
      4) the NVMe layer may request further NVMe or Fabric commands to be processed via the transport Admin Queue connection. The additional commands may perform NVMe Fabrics authentication or may be NVMe or NVMe Fabric commands to get/set properties to configure the newly created NVMe controller instance created by the Admin Queue Connect command.
         i) the commands are classified as no-data (see B.1.1), read (see B.1.2), or write (see B.1.3 or B.1.4) operations.
      5) as this is a NVMe Discovery Service, no IO queues are created.

6) the NVMe layer issues a Get Log Page command, with Log Identifier set to 70h, to read the Discovery Log Entries from the Discovery Service.
   i) the command is classified as a read operation (see B.1.2).
7) the NVMe layer may determine that no further interaction with the Discovery Service is necessary and may use the FC-NVMe layer to terminate the service.
   i) send NVMe_Disconnect LS to the Discovery Service. The LS parameters will indicate to terminate the association.
   ii) the FC-NVMe target receives the LS and generates the LS response.
   iii) the transport association and all connections for it are terminated.
   iv) if this was the only association between the initiator NVMe_Port and target NVMe_Port, the login may be terminated:
      1) send LOGO to the FC-NVMe target.

7) Issue a GID_FF (see FC-GS-8) query to the Name Server with the Domain_ID Scope and Area_ID Scope fields set to zero, the FC-4 Feature Bits field set to 01h (i.e., NVMe over FC target function supported), and the Type code field set to 28h (i.e., NVMe over FC). This query obtains a list of the Port Identifiers (see FC-GS-8) of devices that support the NVMe over FC protocol, and support the NVMe over Fabrics Target Port Function;
8) During operation, if the NVMe layer chooses to communicate with a NVMe (i.e., storage) subsystem identified in one of the Discovery Log records, the NVMe layer uses the FC-NVMe layer to establish a session with the NVMe subsystem:
   i) the initiator NVMe_Port ensures there is connectivity to the target NVMe_Port. The information passed to it from the NVMe layer will minimally indicate the WWNN and WWPN of the target.
      1) interact with the FC Name Server to resolve the WWNN and WWPN and Fabric information to ensure that it has connectivity. A GID_FF query with FC-4 Feature Bits field set to 01h may be used to validate the N_Port supports an NVM subsystem.
      2) if there is connectivity, the initiator acquires the N_Port_ID to use for subsequent communication with the target.
   ii) the initiator NVMe_Port ensures there is a login with the FC target NVMe_Port.

NOTE 1 - Note: if there is already an active login between the NVMe initiator and target N_Port's, these steps may be skipped:

      1) send PLOGI;
      2) send PRLI with Type field set to 28h;
   iii) the NVMe layer interacts with the FC-NVMe layer to create an association and create the initial Admin Queue connection:

      send Create Association NVMe_LS to the NVM subsystem.

   iv) the NVMe layer issues a NVMe Connect command via the newly created transport Admin Queue connection. The Connect command is to create the Admin Queue.
      1) the Connect command is classified as a write operation (see B.1.3 or B.1.4).
   v) the NVMe layer may request further NVMe or NVMe over Fabric commands to be processed via the transport Admin Queue connection. The additional commands may perform NVMe over Fabrics authentication or may be NVMe or NVMe over Fabrics commands to get/set properties to configure the newly created NVMe controller instance created by the Admin Queue Connect command.
      1) The commands are classified as no-data (see B.1.1), read (see B.1.2), or write (see B.1.3 or B.1.4) operations.

vi) the NVMe layer determines the number of IO Queues it wants to create on the NVMe controller. The NVMe layer interacts with the FC-NVMe layer to create one or more IO Queue connections. For each IO Queue connection:

1) send Create Association NVMe_LS to the NVM subsystem.
2) the NVMe layer issues a NVMe Connect command via the newly created transport Admin Queue connection. The Connect command is to create the IO Queue.
   i) the Connect command is classified as a write operation (see B.1.3 or B.1.4).
3) the NVMe layer may perform additional commands on the newly-created IO Queue and connection, such as authentication commands
   i) the commands are classified as no-data (see B.1.1), read (see B.1.2), or write (see B.1.3 or B.1.4) operations.

vii) at this point the NVMe controller is fully operational. The NVMe layer may request further NVMe or NVMe over Fabric commands to be processed via the transport Admin Queue connection or via one of the IO Queue connections.

1) the commands are classified as no-data (see B.1.1), read (see B.1.2), or write (see B.1.3 or B.1.4) operations.

9) At this point the Initiator may terminate the association with the Discovery Controller (see C.1.3).

C.1.2 Initiator discovery of direct-attached target NVMe_Ports (no switched Fabric topology)

The following procedure may be used by initiator NVMe_Ports for discovering NVMe over FC devices in a scenario where no switched Fabric is present. Examples are direct N_Port to N_Port or VN_Port to VN_Port topologies.

Depending on the specific configuration and the management requirements, any of the following steps may be omitted and may be performed using actions outside this standard or the referenced standards.

1) Send PLOGI;
2) Send PRLI send PRLI with Type field set to 28h;
3) If the PRLI did not succeed, the other endpoint does not support FC-NVMe and communication is stopped.
4) If FC-NVMe is supported and the returned Feature bits indicate support for NVMe Discovery Service:
   i) the steps in C.1.1 step 6, i may be followed to create an association with the Discovery Service and obtain the Discovery Log records on the device.
5) During operation, the NVMe layer chooses to communicate with an NVM subsystem identified in one of the Discovery Log records. Therefore, the NVMe layer uses the FC-NVMe layer to establish a session with the NVM subsystem:
   i) the initiator NVMe_Port ensures there is connectivity to the target NVMe_Port. The information passed to it from the NVMe layer will minimally indicate the WWNN and WWPN of the target NVMe_Port.
      1) the WWNN and WWPN must correspond to the other FC endpoint and the Fabric information must correlate to a direct-connection.
      2) if there is connectivity, the initiator shall have the N_Port_ID to use for subsequent communication with the target.

ii) the steps in C.1.1 step 8, ii through step 8, vii may be followed to create an association and enact communication with the NVM subsystem.

6) At this point the Initiator may terminate the association with the Discovery Controller (see C.1.3).

C.1.3 NVMe association termination

The NVMe layer may encounter conditions which causes it to terminated the association. (e.g., error recovery, controller reset, or no longer needing connectivity to the NVM subsystem). If the NVMe layer decides to terminate the association, it uses the FC-NVMe layer to terminate the service by processing the following steps:

1) send Disconnect NVMe_LS to the associated NVMe_Port. The Disconnect NVMe_LS parameters indicate to terminate the association;
2) the target NVMe_Port receives the Disconnect NVMe_LS and generates the Disconnect NVMe_LS response;
3) the transport association and all connections for it are terminated; and
4) if this was the only association between the initiator NVMe_Port and target NVMe_Port, the login may be terminated:
   i) send LOGO to the FC-NVMe target.

C.1.4 Initiator RSCN reception

During operation, the initiator NVMe_Port may receive a RSCN for the N_Port which supports the NVMe Discovery Service:

1) the FC layer processes the RSCN;
2) the FC-NVMe initiator communicates the potential change notice to the NVMe layer; and
3) the NVMe layer may repeat steps in 13.b to obtain an updated Discovery Log from the NVMe Discovery service on the N_Port_ID that generated the state change.

Annex D
(informative)
Error detection and recovery examples

D.1 Overview

This informative annex diagrams various error detection and recovery procedures for NVMe_Ports conforming to this standard. The conventions for the diagrams are shown in table D.1.

Table D.1 - Diagram conventions

Convention	Meaning
⟶	Class 3 frame.
X	Frame lost or dropped.
Initiator	initiator NVMe_Port
Target	target NVMe_Port

An example of a lost NVMe_CMND or lost NVMe_RSP with controller reset is shown in figure D.1.

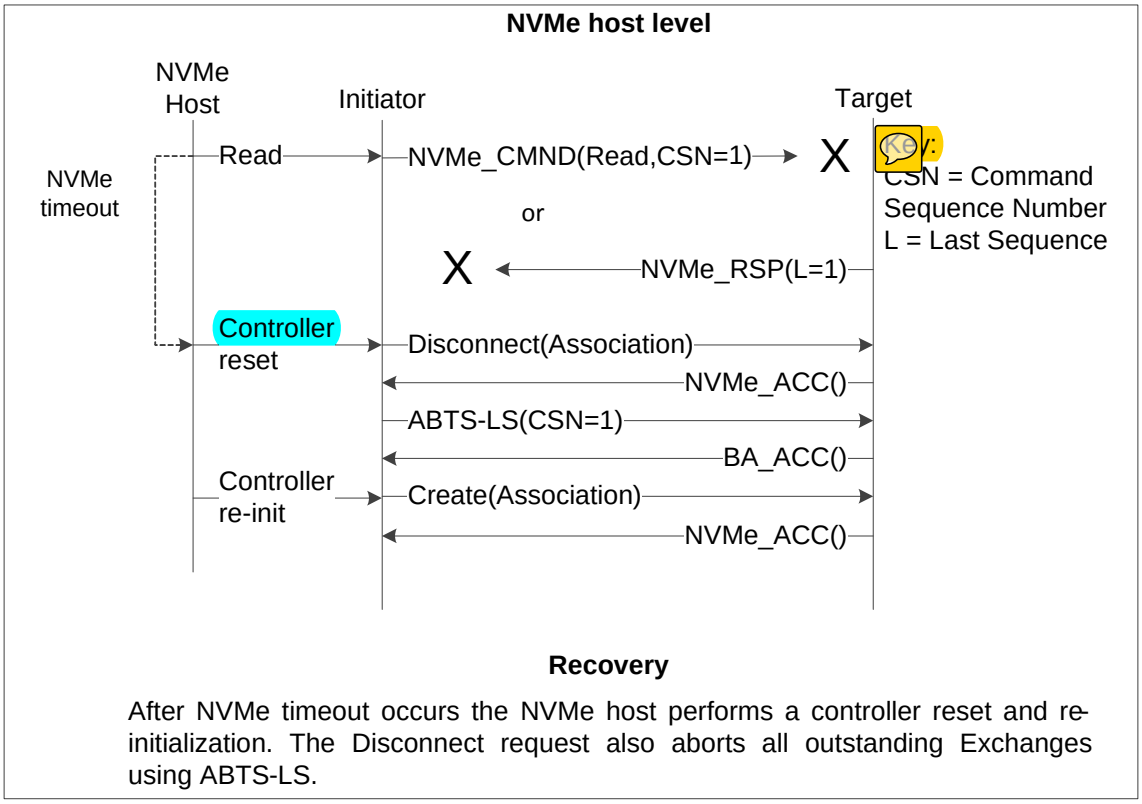


Figure D.1 - NVMe_CMND lost or NVMe_RSP lost with controller reset

example of a lost NVMe_CMND or lost NVMe_RSP with NVM Abort is shown in figure D.2.

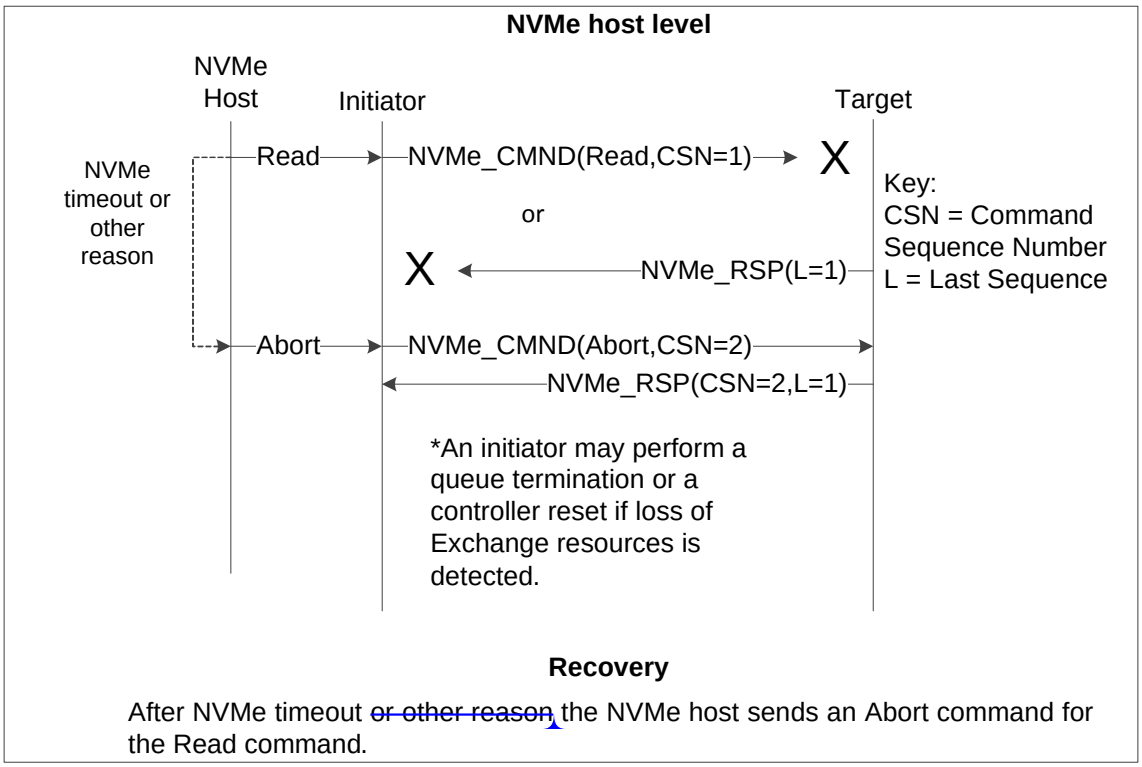


Figure D.2 - NVMe_CMND lost or NVMe_RSP lost with NVM Abort

An example of a lost NVMe_XFER_RDY with NVM Abort is shown in figure D.3.

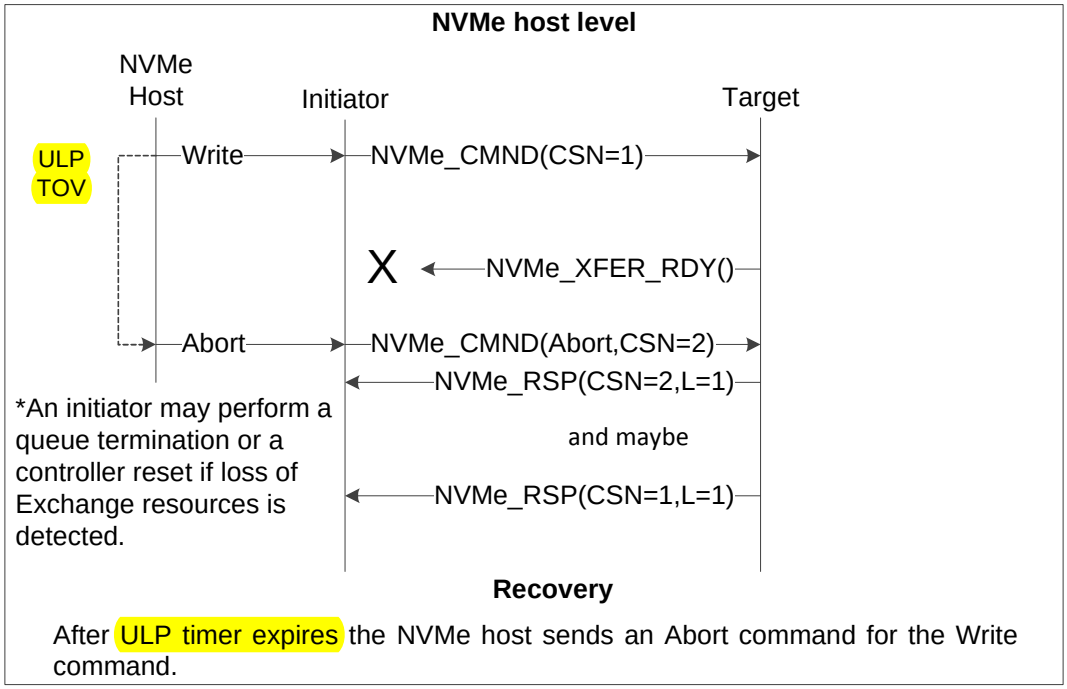


Figure D.3 - NVMe_XFER_RDY lost with NVM Abort