FC-LS-4Copies 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

INCITS 540-201x
T11/Project 540-D/Rev 1.16

FIBRE CHANNEL

NVME

(FC-NVMe)

REV 1.16

INCITS working draft proposed
American National Standard
for Information Technology

July 24, 2017

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
12701 Whitewater Drive
Minnetonka, MN 55343
Voice: 952-687-2431
craig.carlson@qlogic.com

Craig Carlson (FC-NVMe Chair)
QLogic Corporation
12701 Whitewater Drive
Minnetonka, MN 55343
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 1.16
T11-2017-00020-v007

Rev 1.15
T11-2017-00020-v00x

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
16-026v0 - FC-NVMe: Frame_Header
16-021v2 - FC-NVMe: Link Services
16-014v2 - FC-NVMe: FC-4 Link Service
16-103v1 - FC-NVMe: FC-4 Name Server registration and objects
16-043v0 - FC-NVMe: NVMe over Fabrics

Rev 1.02
15-388v0 - FC-NVMe FCP CMD_IU Format
15-442v2 - FC-NVMe IU formats
15-446v2 - FC-NVMe FC-4 Link Service definitions

Rev 1.01 - Revised structure

Rev 1.00 - Initial draft standard

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.

CAUTION: 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 and following calls for the identification of patents that may be required for the implementation of the standard, no such claims have been made. 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 licenses are not required to avoid infringement in the use of this standard.

Published by

American National Standards Institute
25 West 43rd Street, 4th floor 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 2014-2017. The standards approval process started in 2016. 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 over Fabrics specification.

Contents **Page**

Figure **Page**

Table **Page**

American National Standard
for Information Technology —

Fibre Channel —
NVMe (FC-NVMe)

1 Scope

This standard defines a 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 specification.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ANSI INCITS 545:201x, *Fibre Channel - Framing and Signaling - 5 (FC-FS-5)* (under consideration)

ANSI INCITS 547:201x, *Fibre Channel - Switch Fabric - 7 (FC-SW-7)* (under consideration)

ANSI INCITS 553:201x, *Fibre Channel - Link Services - 4 (FC-LS-4)* (under consideration)

ANSI INCITS 548:201x, *Fibre Channel - Generic Services - 8 (FC-GS-8)* (under consideration)

ANSI INCITS 546:201x, *SCSI Architecture Mode - 6 (SAM-6)* (under consideration)

NVM Express revision 1.3 - May 1, 2017

NVMe over Fabrics revision 1.0 - June 5, 2016

3 Definitions, abbreviations, symbols, keywords, and conventions

3.1 Definitions

3.1.1 Association ID
value that uniquely identifies an NVMeoFC association

3.1.2 Connection ID
value that uniquely identifies an NVMeoFC connection

3.1.3 Data Overlay
act of transferring data at the same offset within a Data Series more than once during the processing of an NVMe command

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

3.1.5 Discovery
steps taken by an NVMe host to detect the presence of NVM subsystems (see NVM Express) and obtain sufficient information to initiate the creation of NVMeoFC associations

3.1.6 First Burst
transmission, by the initiator NVMe_Port, of the first NVMe_DATA IU in a Data Series for an NVMeoFC write operation following the transmission of the NVMe_CMND IU

3.1.7 initiator NVMe_Port
NVMe_Port which is the NVM host port for an NVMeoFC association

3.1.8 LBA data
data read from or written to a namespace (see NVM Express)

3.1.9 NVM host port
VN_Port that acts as an interface between an NVMe host and an NVMe-oF fabric (see NVMe over Fabrics)

3.1.10 NVMe command
NVM Express or NVMe-oF fabrics command (see NVMe over Fabrics) issued by an NVMe host to a controller (see NVM Express) via an SQ (see NVM Express)

3.1.11 NVMe host
entity that submits NVMe commands to a controller for execution and receives NVMe command completions from the same controller

3.1.12 NVMe timeout
NVMe host event which occurs if the NVMe host does not receive a CQE (see NVM Express) for an NVMe command within a time duration expected by the NVMe host

3.1.13 NVMe_Port
Nx_Port (see FC-FS-5) that supports the FC-NVMe standard

3.1.14 NVMe-oF controller
implementation of a controller on an NVMe transport (see NVMe over Fabrics) other than PCIe

3.1.15 NVMeoFC association

exclusive communication relationship between an initiator NVMe_Port and a target NVMe_Port for an association (see NVM Express)

3.1.16 NVMeoFC I/O operation

Fibre Channel exchange that is uniquely associated with an NVMe command

3.1.17 read operation

NVMe command which transfers data from a controller to an NVMe host

3.1.18 scatter/gather list (SGL)

list consisting of <address, length> pairs which describe the locations in NVMe host memory that are to be used for NVMe command data transfers

3.1.19 SGL data

data that will be read from or written to the memory locations described by the <address, length> pairs contained in an SGL

3.1.20 target NVMe_Port

NVMe_Port which is the NVM subsystem port (see NVMe over Fabrics) for an NVMeoFC association

3.1.21 write operation

NVMe command which transfers data from an NVMe host to a controller

3.2 Abbreviations

Abbreviations and acronyms applicable to this standard are listed. Definitions of several of these items are included in 3.1.

ABTS Abort Sequence
ABTS-LS ABTS Abort Exchange
BLS Basic Link Service
CQ Completion Queue
CQE Completion Queue Entry
ELS Extended Link Service
FC Fibre Channel
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
LBA Logical Block Address
LOGO N_Port Logout
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
NQN NVMe Qualified Name
NVMe™ NVM Express
NVMeoFC NVM Express over Fibre Channel

NVMe-oF™ NVM Express over Fabrics
NVMe_LS NVMe FC-4 Link Service
PLOGI N_Port Login
PRLI Process Login
PRLO Process Logout
SGL Scatter/gather List
SQ Submission Queue
SQE Submission Queue Entry
TPRLO Third Party Process Logout

3.3 Symbols

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

!= not equal

3.4 Keywords

3.4.1 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.

3.4.2 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.

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

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

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

3.4.6 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.

3.4.7 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.

3.4.8 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.

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

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

3.5 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.

4 General

4.1 Structure and concepts

Fibre Channel (FC) is logically a point-to-point serial data channel. The Fibre Channel Physical layer (i.e., FC-2 layer) described by FC-FS-5 performs those functions required to transfer data from one Nx_Port to another. In this standard, Nx_Ports capable of supporting NVM Express over FC (NVMeoFC) 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.

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 types of functional management:

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

The NVMeoFC protocol defines the mapping of NVMe over Fabrics (NVMe-oF) 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 NVMeoFC is mapped into a Fibre Channel Exchange. A Fibre Channel Exchange carrying information for an NVM Express over Fabrics I/O operation is an NVMeoFC Exchange. The requests and responses of an I/O operation are mapped into Information Units (IUs) as specified in table 31 and table 32.

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.

NVMeoFC protocol layers are shown in figure 1.

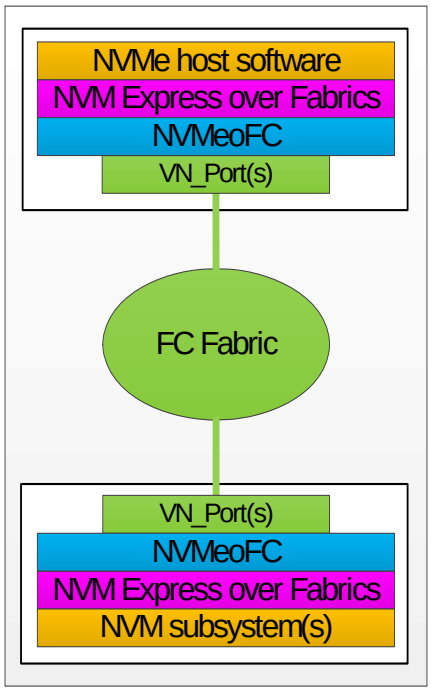

Figure 1 – NVMeoFC protocol layers

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 (see NVMe over Fabrics), and the interfaces to the NVMe host software and NVM subsystem(s) protocol layer are specified in the NVM Express specification (see NVM Express).

The NVMeoFC target device functional model is shown in figure 2.

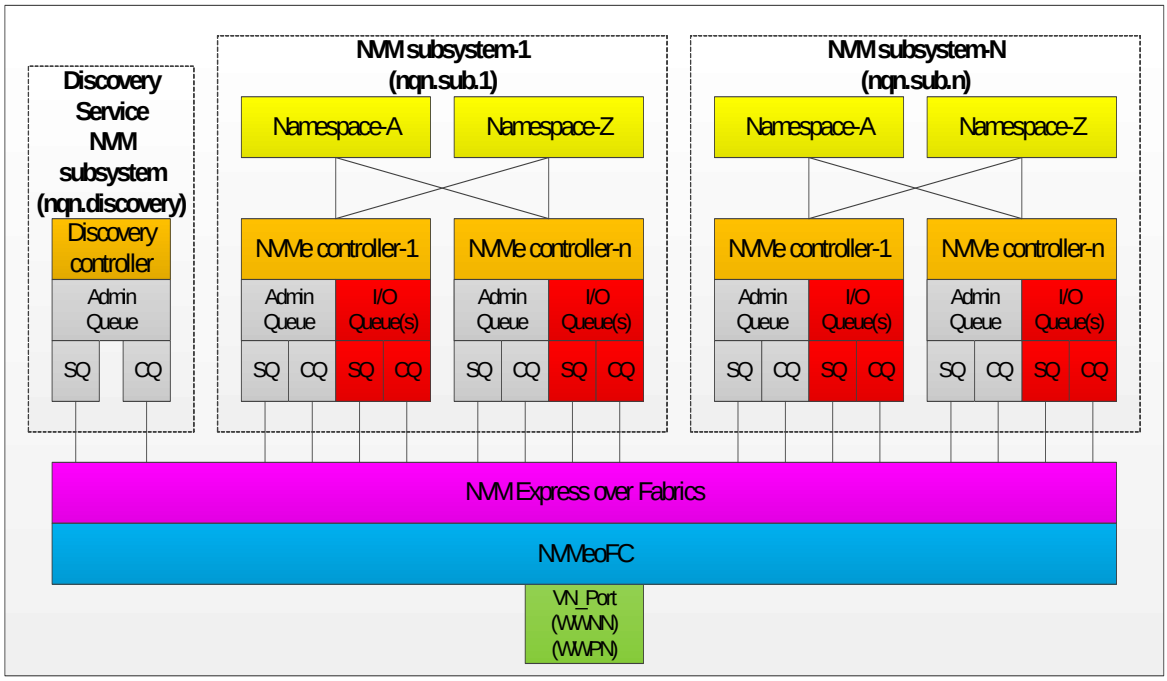

Figure 2 – NVMeoFC target device functional model

As shown in figure 2, the NVM Express over Fabrics layer interfaces to a Discovery Service NVM subsystem (see NVMe over Fabrics) and one or more NVM subsystems (see NVM Express).

A Discovery Service NVM subsystem consists of a Discovery controller with an Admin Queue and associated Submission Queue (SQ) and Completion Queue (CQ).

An NVM subsystem consists of one of more NVMe controllers, each with an Admin Queue and associated Submission Queue (SQ) and Completion Queue (CQ), and one or more I/O Queues and associated Submission Queue (SQ) and Completion Queue (CQ).

4.2 NVMeoFC ports

A target NVMe_Port corresponds to a physical interface connecting one or more a NVM subsystems to an FC Fabric. A target NVMe_Port provides FC Fabric connectivity for one or more NVM subsystems. For a particular NVM subsystem, the functions of an NVM subsystem port are provided by a target NVMe_Port. As NVM subsystem Port IDs (see table 39) 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 provides FC Fabric connectivity for one or more NVMe hosts.

4.3 NVMeoFC association

An NVMeoFC association is an NVMeoFC layer 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).

The NVMeoFC association is created by transmitting a Create_Association NVMe_LS request (see 8.5). If the target NVMe_Port and NVM subsystem allow the communication relationship to be created, the target NVMe_Port transmits a Create_Association NVMe_LS accept payload (see 8.5) to the initiator NVMe_Port. The Create_Association accept payload contains an association identifier that shall be used by the NVMeoFC layer on the initiator NVMe_Port to refer to the NVMeoFC association in subsequent Fabric traffic transmitted to the target NVMe_Port. If the NVMeoFC association cannot be created, the target NVMe_Port shall transmit an NVMe_RJT (see 8.3) 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.

A related NVMe over Fabrics association is created by the first NVMe over Fabrics Connect command (see NVMe over Fabrics), issued on the association's Admin Queue connection, to create the Admin Queue. Refer to NVMe over Fabrics for additional requirements and behaviors of NVMe over Fabrics associations and Admin Queue creation.

An active NVMeoFC association is terminated if:

a) any NVMeoFC connection for the NVMeoFC association is terminated;
b) a Disconnect NVMe_LS (see 8.7) is received;
c) one of the clearing effects (see 4.13) occur which terminates the NVMeoFC association; or
d) the NVMe host or NVM subsystem detects a condition which causes it to terminate the corresponding NVMe-oF association.

Explicit termination of an NVMeoFC association is requested by the transmission of a Disconnect NVMe_LS.

The initiator NVMe_Port or the target NVMe_Port may decide to terminate an NVMeoFC association. If an NVMe_Port decides to terminate an NVMeoFC association, the NVMe_Port shall explicitly request termination of the NVMeoFC association by transmitting a Disconnect NVMe_LS unless there is no longer a valid login. The NVMe_Port may proceed with the termination of the association without waiting for either an NVMe_ACC or NVMe_RJT status for the Disconnect NVMe_LS.

All initiator NVMe_Ports and target NVMe_Ports shall be capable of transmitting, as well as accepting and processing, a Disconnect NVMe_LS request and response.

If an NVMeoFC association is terminated, the NVMeoFC layer on the initiator NVMe_Port or target NVMe_Port shall implicitly terminate all Admin Queue and I/O Queue connections for the association.

If an NVMe_Port receives a NVMe_LS request that does not correspond to an active NVMeoFC association, then the 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).

Other than Name Server registration (see 7), the manner in which the NVMe host discovers possible NVM subsystems is outside the scope of this standard.

4.4 NVMeoFC connection

An NVMeoFC connection is an NVMeoFC layer abstraction representing an NVMe Submission Queue (SQ) (see NVM Express) and an NVMe Completion Queue (CQ) (see NVM Express) for an

NVMe controller (see NVM Express). An NVMeoFC connection may correspond to the controller's Admin Queue or an I/O Queue on that controller.

An NVMeoFC connection corresponding to the Admin Queue is created simultaneously with the creation of the NVMeoFC association as part of the processing of the Create Association NVMe_LS request (see 8.5). Successful creation of the NVMeoFC association shall also include allocation of the resources for the NVMeoFC connection for the controller's Admin Queue (i.e., SQ and CQ). The Create Association NVMe_ACC payload (see 8.5) contains a Connection ID that is used by the initiator NVMe_Port and target NVMe_Port to refer to the NVMeoFC connection for the controller's Admin Queue. The tuple <Connection ID, initiator NVMe_Port N_Port_ID, target NVMe_Port N_Port_ID> shall be unique.

The first NVMe command on the NVMeoFC connection is the NVMe-oF Connect command (see NVMe over Fabrics) for the NVMe controller's Admin Queue. See NVMe over Fabrics for additional requirements and behaviors of Admin Queue creation.

An NVMeoFC connection corresponding to an I/O Queue is created when the NVMe host makes a request of the NVMeoFC layer to establish the transport connection for an I/O Queue for a particular controller. The NVMeoFC layer initiates the creation of the transport connection by transmitting a Create I/O Connection NVMe_LS request (see 8.6) from the initiator NVMe_Port to the target NVMe_Port.

If the target NVMe_Port and NVMe controller accepts the request, the target NVMe_Port transmits a Create I/O Connection NVMe_ACC payload (see 8.6) to the initiator NVMe_Port. The Create I/O Connection NVMe_ACC payload contains a Connection ID that is used by the initiator NVMe_Port and target NVMe_Port to refer to the NVMeoFC connection for the controller I/O queue specified by the request. The tuple <Connection ID, initiator NVMe_Port N_Port_ID, target NVMe_Port N_Port_ID> shall be unique.

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 NVMeoFC 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 NVMeoFC association is valid and the NVMeoFC connection cannot be created, then 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 NVMe command on an NVMeoFC connection for an I/O Queue is the NVMe-oF Connect command for the corresponding NVMe controller I/O Queue. See NVMe over Fabrics for additional requirements and behaviors of I/O Queue creation.

Once established, the NVMeoFC connection remains in place until the NVMeoFC association is terminated, or a transport error occurs that causes loss of a message or loss of data that the NVMeoFC layer is unable to recover.

The initiator NVMe_Port or the target NVMe_Port may terminate an NVMeoFC connection. If an NVMe_Port terminates an NVMeoFC connection, the NVMe_Port shall transmit a Disconnect NVMe_LS.

The termination of an NVMeoFC connection terminates the NVMeoFC association.

If an NVMeoFC 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). Thus, the termination of the NVMeoFC connection shall cause the

NVMeoFC layer on the initiator NVMe_Port and target NVMe_Port to implicitly terminate all outstanding NVMeoFC I/Os that are associated with the NVMeoFC connection.

If an NVMeoFC connection is terminated at an NVMe_Port, then for each outstanding NVMeoFC I/O operation (see 4.5) on the connection, that NVMe_Port shall transmit an ABTS-LS to terminate the Exchange.

If an NVMe_Port receives an NVMe_LS request that contains an unknown Connection ID, the NVMe_Port shall not process the NVMe_LS and the 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 a target NVMe_Port receives an NVMe_CMND IU that contains an unknown Connection ID, or an initiator NVMe_Port receives an NVMe_CMND IU, the receiving NVMe_Port should transmit an ABTS-LS for the corresponding Exchange.

If an initiator NVMe_Port receives a Create Association NVMe_LS or a Create Connection NVMe_LS, the initiator NVMe_Port shall transmit an NVMe_RJT with the reason code set to 07h (i.e., Protocol Error) and the reason code explanation set to 00h (i.e., No additional explanation).

4.5 NVMeoFC 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 (see figure 1) which then submits the command to the NVMeoFC layer. The NVMeoFC layer specifies the NVMeoFC association (see 4.3) with the NVMe controller, and the NVMeoFC connection (see 4.4) for the SQ with which the command is associated, and delivers the command to the initiator NVMe_Port.

The initiator NVMe_Port allocates an Exchange resource for the NVMeoFC I/O operation and associates the NVMe command in the SQE to the Exchange. All NVMe IUs for the NVMeoFC I/O operation shall be transmitted as part of that Exchange. The initiator NVMe_Port creates a NVMe_CMND IU (see 9.2) 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 NVMeoFC connection (i.e., SQ). The NVMe_CMND IU specifies the Connection Identifier for the NVMeoFC connection (i.e., the NVMe controller and the Queue ID), to which the SQE is being submitted.

The initiator NVMe_Port transmits the NVMe_CMND IU payload to start the NVMeoFC I/O operation. The Exchange that is started is identified by its fully qualified X_ID (see FC-FS-5) during the remainder of the NVMeoFC I/O operation and is used only for the IUs associated with that NVMeoFC I/O operation.

Data transfer for the NVMeoFC I/O operation occurs as specified in 9.4. Upon completion of command processing and data transfer, if any, is completed, the target NVMe_Port shall transmit a response IU. The response IU conveys an NVMe CQE which indicates the completion status of the NVMe command. The response IU shall be a NVMe_RSP IU (see 9.5) or NVMe_ERSP IU (see 9.6).

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

If a response IU is received by the initiator NVMe_Port and the target NVMe_Port requested confirmed completion (see 4.10), then the initiator NVMe_Port shall transmit an NVMe_CONF IU (see 9.7) to close the Exchange.

Upon reception of a response IU with no NVMeoFC error, the initiator NVMe_Port shall deliver the CQE received, or implied by the response IU, to the NVMe CQ associated with the NVMeoFC connection that was specified in the NVMe_CMND IU.

If an NVMe_Port receives an NVMeoFC frame in a service other than Class 3, then the NVMe_Port shall discard the frame.

4.6 First burst

The First Burst Supported bits in the PRLI ELS request NVMeoFC Service Parameter page (see 6.3.2) and PRLI ELS accept NVMeoFC Service Parameter page (see 6.3.3) are used to determine the support for first burst.

If both the initiator NVMe_Port and target NVMe_Port support first burst (see table 6), then the initiator NVMe_Port may choose to perform write operations by sending a first NVMe_DATA IU (see 9.4) without a preceding NVMe_XFER_RDY IU (see 9.3). 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 data for a write operation. 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 the data for a write operation originally sent in the first NVMe_DATA IU as well as for any remaining data for a write operation. The initiator NVMe_Port shall support retransmission of the data for a write operation originally sent in the first NVMe_DATA IU.

If the initiator NVMe_Port or the target NVMe_Port do not support first burst, then the initiator NVMe_Port shall not send an NVMe_DATA IU without having received a preceding NVMe_XFER_RDY IU, and the target NVMe_Port shall transmit one or more NVMe_XFER_RDY IUs requesting each NVMe_DATA IU to perform the write operation.

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.

NVMe commands that are part of fused operations (see NVM Express) are required to be processed in the order they were sent by the initiator. To allow these commands to be processed in the order they were sent, if the order was not maintained by the Fabric, the Command Sequence Number is used (see 4.7.2).

All NVMe_ERSPs are required to be processed in the order that they were sent. To allow the NVMe_ERSP IU to be processed in order, if 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. For an NVMe_RSP, a CQE shall be generated as specified in 4.8.2 and sent to the NVMe host.

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. Separate incrementing counters are maintained for each NVMeoFC connection. The following rules specify how to use the CSN:

a) the CSN shall be equal to zero for the first NVMe_CMND IU for each NVMeoFC connection and shall be incremented by one for each subsequent command on the NVMeoFC connection;
b) the CSN shall wrap from FFFF FFFFh to zero;
c) the CSN reflects the order that the SQE was submitted to the Submission Queue; and
d) the target NVMe_Port shall use the CSN to place the commands into the target Submission Queue in the proper order for any commands that are required to be in-order (i.e., fused operations (see NVM Express)).

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. Separate incrementing counters are maintained for each NVMeoFC connection. The following rules specify how to use the RSN:

a) the RSN shall be equal to zero for the first NVMe_ERSP IU (see 9.6) for each NVMeoFC connection and shall be incremented by one for each subsequent NVMe_ERSP IU on the NVMeoFC connection;
b) the RSN shall wrap from FFFF FFFFh 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

An NVMe_RSP IU may be sent by a target NVMe_Port with the following exceptions:

a) at least one 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 SQ Head Pointer (SQHD) (see NVM Express) to be transmitted periodically;
b) an NVMe_ERSP IU shall be sent by the target NVMe_Port if the SQ is 90% or more full (see NVM Express);
c) an NVMe_ERSP IU shall be sent by the target NVMe_Port for responses to any command that is part of a fused command pair (i.e., is part of a fused operation);
d) an NVMe_ERSP IU shall be sent by the target NVMe_Port if the NVM 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);
e) an NVMe_ERSP IU 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 IU may be sent by the target NVMe_Port for any other reason 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 with:

a) the SQHD set to the SQHD value of the CQE in the last NVMe_ERSP IU that was delivered to the NVMe host;
b) the Command_ID set to the value sent in the associated 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 ERSP Result field value of 00h (i.e., SUCCESS) and the Transferred Data Length field value (see 9.6) is equal to the initiator NVMe_Port's transfer byte count (see 9.4.3), then the command shall be completed by delivery of the CQE contained within the NVMe_ERSP IU to the NVMe-oF layer.

If a command is completed by an NVMe_ERSP IU with a ERSP Result field value other than 00h (i.e., not SUCCESS) or with a Transferred Data Length field value that is not equal to the initiator NVMe_Port's transfer byte count, then the NVMeoFC layer shall not deliver the CQE to the NVMe-oF layer, but shall communicate the detected error to the NVMe layer. The manner in which the NVMeoFC layer communicates the error to the NVMe-oF layer, and the resulting recovery actions, is outside the scope of this standard.

4.10 Confirmed completion of NVMeoFC I/O operations

Some implementations require an acknowledgment of successful delivery (i.e., confirmed completion) of NVMe_RSP IU or NVMe_ERSP IU information. Such an acknowledgment is provided by requesting an NVMe_CONF IU. The Confirmed Completion Supported bits in the PRLI ELS request NVMeoFC Service Parameter page (see 6.3.2) and PRLI ELS accept NVMeoFC Service Parameter page (see 6.3.3) are used to determine the support for confirmed completion.

If an initiator NVMe_Port and a target NVMe_Port both indicate support of confirmed completion (see table 6), then a target NVMe_Port may request an NVMe_CONF IU by setting the Last_Sequence bit to zero (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 the Last_Sequence bit set to one 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.

4.11 Data transfer

4.11.1 Overview

NVMe over Fabrics (see NVMe over Fabrics) 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 specified by SGL(s) in a command capsule and is data that is not transfered within a command or response capsule, but by using a transport specific data transfer mechanism. NVMeoFC rules are specified in 4.11.2 and 4.11.3 for the following types of NVMe-oF data transfers:

a) in-capsule data;
b) SGL data for write operations; and
c) SGL data for read operations.

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).

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., transmission of data referenced by SGLs, see NVMe over Fabrics), 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 NVMeoFC:

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 of zero, indicating the first byte of data represented by the SGL, and the length of the data (see 4.11.3.3).

An SGL example is shown in figure 3.

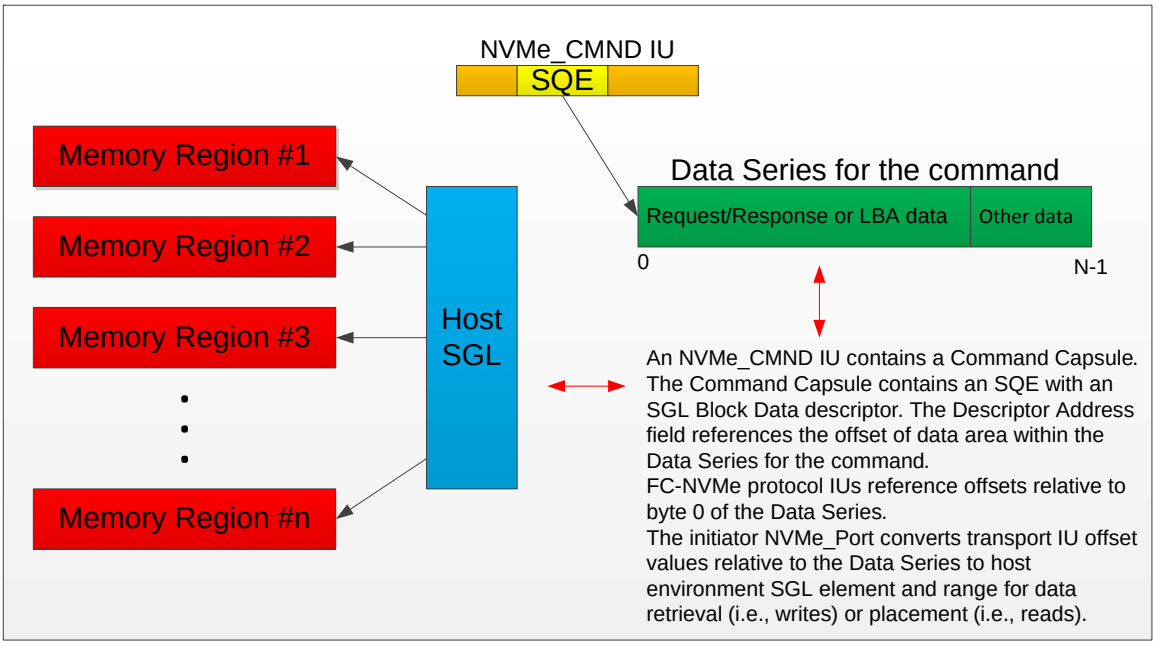

Figure 3 – SGL example

4.11.3.3 SGL entry format

The SGL within a transmitted SQE (see NVMe over Fabrics) shall be set as follows:

a) the SGL Descriptor Type field shall be set to 0h (i.e., SGL Data Block descriptor);
b) the SGL Descriptor Sub Type field shall be set to 0h (i.e., Address);
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 NVMeoFC capabilities

Some NVMeoFC capabilities require negotiation between the initiator NVMe_Port and target NVMe_Port and the initiator NVMe_Port for such capabilities to be used. NVMeoFC capabilities are:

a) initiator NVMe_Port;
b) target NVMe_Port;
c) Discovery Service;
d) Confirmed Completion Supported; and
e) First Burst Supported.

Discovery of these capabilities is provided by Process Login (see 6.3).

4.13 Clearing effects of NVMeoFC, FC-FS-5, and FC-LS-4 actions

Table 1 and table 2 summarize the clearing effects resulting from Fibre Channel link actions and NVMeoFC operations, respectively. The clearing effects are applicable only to Exchanges associated with NVMeoFC operations.

Clearing effects of target power cycle and link related actions are specified in table 1.

Table 1 – Clearing effects of target power cycle and link related actions

Clearing effect	Target Power Cycle	FC link action			
		LOGO ELS[b], PLOGI ELS[d]	PRLI ELS PRLO ELS[b,e],	TPRLO ELS[a]	ABTS-LS
PLOGI ELS parameters set to default values (see FC-LS-4)					
For all logged-in initiator NVMe_Ports	Y	N	N	N	N
Only for initiator NVMe_Port associated with the action	-	Y	N	N	N
Active NVMeoFC Associations terminated					
For all initiator NVMe_Ports	Y	N	N	Y	N
Only for initiator NVMe_Port associated with the action	-	Y	Y	-	N
Only for NVMeoFC Association associated with the action	-	-	-	-	Y
Active NVMeoFC Connections terminated					
For all initiator NVMe_Ports	Y	N	N	Y	N
Only for initiator NVMe_Port associated with the action	-	Y	Y	-	N
Only for NVMeoFC Association associated with the action	-	-	-	-	Y

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.
d) If an LS_ACC is received, then the new PLOGI parameters take effect.
e) If an LS_ACC is received, then the new PRLI parameters take effect.

Table 1 – Clearing effects of target power cycle and link related actions (Continued)

Clearing effect	Target Power Cycle	FC link action			
		LOGO ELS[b], PLOGI ELS[d]	PRLI ELS PRLO ELS[b,e],	TPRLO ELS[a]	ABTS-LS
Open NVMeoFC Exchanges terminated					
For all initiator NVMe_Ports	Y	N	N	Y	N
Only for initiator NVMe_Port associated with the action	-	Y	Y	-	N
Only for NVMeoFC Association associated with the action	-	-	-	-	Y
Only for NVMeoFC Connection associated with the action	-	-	-	-	-
Only for NVMeoFC Exchange associated with ABTS	-	-	-	-	-
Process Login parameters cleared[c]					
For all logged-in initiator NVMe_Ports	Y	N	N	Y	N
Only for NVMe_Port associated with the action	-	Y	Y	-	N
CSN set to zero					
For all initiator NVMe_Ports	Y	N	N	Y	N
Only for initiator NVMe_Port associated with the action	-	Y	Y	-	N

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.
d) If an LS_ACC is received, then the new PLOGI parameters take effect.
e) If an LS_ACC is received, then the new PRLI parameters take effect.

Clearing effects of initiator NVMe_Port actions are specified in table 2.

Table 2 – Clearing effects of initiator NVMe_Port actions

Clearing effect	Initiator NVMe_Port action	
	Controller Reset or Shutdown	DISCONNECT NVME_LS
PLOGI ELS parameters set to default values (see FC-LS-4)		
For all logged-in initiator NVMe_Ports	N	N
Only for initiator NVMe_Port associated with the action	N	N
Active NVMeoFC Associations terminated		
For all initiator NVMe_Ports	N	N
Only for initiator NVMe_Port associated with the action	N	N
Only for NVMeoFC Association associated with the action	Y	Y

Table 2 – Clearing effects of initiator NVMe_Port actions (Continued)

Clearing effect	Initiator NVMe_Port action	
	Controller Reset or Shutdown	DISCONNECT NVME_LS
Active NVMeoFC Connections terminated		
For all initiator NVMe_Ports	N	N
Only for initiator NVMe_Port associated with the action	N	N
Only for NVMeoFC Association associated with the action	Y	Y
Only for NVMeoFC Connection associated with the action	-	-
Open NVMeoFC Exchanges terminated		
For all initiator NVMe_Ports	N	N
Only for initiator NVMe_Port associated with the action	N	N
Only for NVMeoFC Association associated with the action	Y	Y
Only for NVMeoFC Connection associated with the action	-	-
Process Login parameters cleared		
For all logged-in initiator NVMe_Ports	N	N
Only for NVMe_Port associated with the action	N	N

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) The Controller Reset function (see NVM Express).

4.14 Port Login and Port Logout

The N_Port Login (PLOGI) ELS is used to establish the Fibre Channel operating parameters between any two Fibre Channel ports, including NVMe_Ports. Implicit login functions are not 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-4. 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 NVMeoFC operating relationships between two NVMe_Ports (see 6.3). The Process Logout (PRLO) ELS request is used to disestablish the NVMeoFC operating relationships between two NVMe_Ports (see 6.4). Implicit PRLI functions are not allowed.

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 NVMeoFC devices (see FC-FS-5 and FC-LS-4).

4.17 NVMeoFC addressing and Exchange identification

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

Each NVMeoFC association is identified by the Association Identifier created by a successful Create Association NVMe_LS request (see 8.5). The Association Identifier is valid as long as the NVMeoFC association is active.

Each NVMeoFC connection is identified by the Connection Identifier created by a successful Create Association NVMe_LS request or Create I/O Connection NVMe_LS request (see 8.6). The Connection Identifier is valid as long as the NVMeoFC connection is active. The NVMe Queue used by the I/O operation is indicated by the Connection Identifier contained in the NVMe_CMND IU issued as the first Sequence of the Exchange.

Each NVMeoFC I/O operation is identified by the NVMeoFC 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 NVMeoFC I/O operations internal to the host and the controller is not defined by this standard.

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 FC-FS-5 Frame_Header is specified in table 3.

Table 3 – FC-FS-5 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 FC-FS-5 Frame_Header are specified in FC-FS-5. The following explanations of the fields provide information about the use of those fields to implement NVMeoFC 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 31 and table 32.

D_ID: The value in the D_ID field is the D_ID of the frame. For NVMeoFC 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 NVMeoFC protocol.

S_ID: The value in the S_ID field is the S_ID of the frame. For NVMeoFC 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 NVMeoFC Exchanges using IUs specified in table 31 and table 32; or
b) 28h (i.e., NVMe over Fibre Channel) (see FC-FS-5) for NVMe_LS Exchanges (e.g., NVMeoFC Link Service requests and responses (see 6) and NVMeoFC FC-4 Link Service requests and responses (see 8)).

Parameter: For a frame with the R_CTL field set to 01h (i.e., solicited data) (i.e., an NVMe_DATA IU), the Parameter field shall contain a relative offset. The relative offset present bit of the F_CTL field shall be set to one, indicating that the Parameter field value is a relative offset. The relative offset shall have a value that is a multiple of 4 (i.e., each frame of each NVMe_DATA IU shall begin on a word boundary).

For a frame with the R_CTL field other than 01h, the relative offset present bit of the F_CTL field shall be set to zero and the Parameter field shall contain a value of zero.

6 NVMeoFC Link Services

6.1 Overview of Link Service requirements

The NVMeoFC link-level protocol includes the Basic Link Services (see FC-FS-5) and Extended Link Services (see FC-LS-4), and the PRLI NVMeoFC 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. NVMeoFC devices introduced into or removed from 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 (see FC-LS-4).

An initiator NVMe_Port shall transmit an explicit PRLI ELS request.

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_LS or NVMeoFC IUs are exchanged. Any NVMeoFC 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 NVMeoFC protocol are defined in 6.3.2 and 6.3.3.

Processing of a PRLI ELS or PRLO ELS request performs the clearing actions defined in table 1 and table 2.

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-4).

A Process Login is successfully completed between two NVMe_Ports only if one port indicates support for the Initiator Function and the other port indicates support for the Target Function. Some capabilities require support by both the initiator NVMe_Port and target NVMe_Port before they may be used.

An accept response code indicating other than ‘Request executed‘ (see 6.3.3 and FC-LS-4) shall be provided if the PRLI ELS NVMeoFC 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-4.

After the completion of any Process Login, all clearing actions specified in table 1 and table 2 shall be performed.

6.3.2 PRLI ELS request NVMeoFC Service Parameter page format

The NVMeoFC Service Parameter page for the PRLI ELS request is specified in table 4.

Table 4 – PRLI ELS request NVMeoFC 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 5.

Table 5 – 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 6.

Table 6 – 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.

Table 6 – Service Parameter Information field format - request and accept(Continued)

Bit	Description
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.
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 4.6). If set to zero, then first burst is not supported.

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 NVMeoFC Service Parameter page are invalid. A Responder receiving such an invalid NVMeoFC 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 NVMeoFC Service Parameter page shall not perform NVMeoFC protocol operations with the Responder.

6.3.3 PRLI ELS accept NVMeoFC Service Parameter page format

The NVMeoFC Service Parameter page for the PRLI ELS accept is shown in table 7.

Table 7 – PRLI ELS accept NVMeoFC 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	

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 8.

Table 8 – 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-4.
7 to 0	Reserved

Service Parameter Information: The format of the Service Parameter Information field is specified in table 6.

First Burst Size: If first burst is not supported (see table 8), then the First Burst Size field shall be set to zero and ignored by the recipient.

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 units of 512 bytes (i.e., a value of one means 512 bytes, two means 1024 bytes).

6.4 PRLO ELS

6.4.1 Overview

The format for the PRLO ELS request and PRLO ELS accept is specified in FC-LS-4.

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.

After Process Logout, no further NVMeoFC communication is possible between those Nx_Ports.

6.4.2 PRLO ELS request NVMeoFC Logout Parameter page format

The NVMeoFC Logout Parameter page for the PRLO ELS request is specified in table 9.

Table 9 – PRLO ELS request NVMeoFC Logout Parameter page

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	TYPE Code	TYPE Code Extension	Reserved	
1	Reserved			
2	Reserved			
3	Logout Service Parameters			

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

TYPE Code Extension: Shall be set to 00h.

Logout Service Parameters: Shall be set to zero.

6.4.3 PRLO ELS accept NVMeoFC Logout Parameter Response page format

The NVMeoFC Logout Parameter Response page for the PRLO ELS accept is specified in table 10.

Table 10 – PRLO ELS request NVMeoFC Logout Parameter Response page

Bits Word	31 .. 24	23 .. 16	15 .. 12	11 .. 08	07 .. 00
0	TYPE Code	TYPE Code Extension	Reserved	Response Code	Reserved
1	Reserved				
2	Reserved				
3	Reserved				

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

TYPE Code Extension: Shall be set to 00h.

Response Code: Shall be set to 0001b.

7 FC-4 Name Server registration and objects

7.1 Overview of FC-4 specific objects for NVMeoFC

The Name Server for a Fibre Channel Fabric is specified in FC-GS-8. NVMeoFC 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 NVMeoFC

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 an NVM subsystem that is capable 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 an NVM subsystem that is capable of operating as a Discovery Service.
1	NVMeoFC initiator function supported. If the NVMeoFC initiator function bit is set to one, then the NVMe_Port is indicating that it has the capability of operating as an NVMeoFC initiator. If the NVMeoFC initiator function bit is set to zero, then the NVMe_Port does not have the capability of operating as an NVMeoFC initiator.
0	NVMeoFC target function supported. If the NVMeoFC target function bit is set to one, then the NVMe_Port is indicating that it has the capability of operating as an NVMeoFC target. If the NVMeoFC target function bit is set to zero, then the NVMe_Port does not have the capability of operating as an NVMeoFC target.

8 NVMeoFC FC-4 Link Services

8.1 Overview

FC-4 Link Service functionality is specified in FC-LS-4. For NVMeoFC 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 shall be a single frame Sequence, and an NVMe FC-4 Link Service response shall be a single frame Sequence, unless otherwise specified.

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.

The NVMe_LS requests and responses are specified in table 12.

Table 12 – NVMe_LS requests and responses

Value (Bits 31-24)	Request/Response	Description	Abbr.	Reference
01h	Response	NVMe_LS reject	NVMe_RJT	8.3
02h	Response	NVMe_LS accept	NVMe_ACC	8.4
03h	Request	Create Association	CASS	8.5
04h	Request	Create I/O Connection	CIOC	8.6
05h	Request	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	Link Service Reject	8.2.3
0000 0003h	Create Association	8.2.4
0000 0004h	Create I/O Connection	8.2.5
0000 0005h	Disconnect	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 Request 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 Link Service Reject descriptor

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

Table 15 – Link Service 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 payload that follows.

reason code: The reason code field contains a value specified in table 16.

Table 16 – NVMe_LS reject reason codes

Value	Description	Meaning
01h	Invalid NVMe_LS command code	The NVMe_LS command code is invalid.
03h	Logical error	The request identified by the NVMe_LS command code and payload is logically inconsistent for the conditions present.
09h	Unable to perform command request	The recipient of the NVMe_LS request is unable to perform the request at this time.
0Bh	Command not supported	The recipient of the NVMe_LS request does not support the command requested.
40h	Invalid Association ID	The Association ID specified in the NVMe_LS request is not a valid ID.
41h	Invalid Connection ID	The Connection ID specified in the NVMe_LS request is not a valid ID.
FFh	Vendor specific	The vendor specific error bits in word 2, bits 7:0 (see table 15) may be used by vendors to specify additional reason codes.
All others	Reserved	

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

Table 17 – NVMe_LS reason code explanations

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

8.2.4 Create Association descriptor

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

Table 18 – Create Association 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)			
17				
18	Host NVMe Qualified Name (HOSTNQN)			
81				
82	NVM Subsystem NVMe Qualified Name (SUBNQN)			
145				
146	Reserved			
255				

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 approximately ten percent of the Submission

Queue Size field value (e.g., send an NVMe_ERSP every NVMe_ERSP Ratio field value completions). A value of zero shall be interpreted as a value of one.

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

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

Host Identifier: contains the Host Identifier as specified in the NVMe over Fabrics Connect Command Capsule.

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

NVM Subsystem NVMe Qualified Name: contains the NVM subsystem NQN as specified in the NVMe over Fabrics 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 NVM I/O queue identifier corresponding to the NVM I/O Queue (see NVM Express) to be created.

Submission Queue Size: contains the number of entries 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 approximately ten percent of the Submission Queue Size field value (e.g., send an NVMe_ERSP every NVMe_ERSP Ratio field value completions).

8.2.6 Disconnect descriptor

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

Table 20 – Disconnect descriptor

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	Descriptor tag = 0000 0005h			
1	Descriptor length			
2	Reserved			
3	Reserved			
4	Reserved			
5	Reserved			

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

8.2.7 Connection Identifier descriptor

The format of the Connection Identifier descriptor is specified in table 21.

Table 21 – Connection Identifier descriptor

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	Descriptor tag = 0000 0006h			
1	Descriptor length			
2	MSB	Connection Identifier		
3				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 NVMoeFC connection.

8.2.8 Association Identifier descriptor

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

Table 22 – 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 uniquely identifies the NVMeoFC association. The tuple <Association Identifier, initiator NVMe_Port N_Port_ID, target NVMe_Port N_Port_ID> shall be unique.

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 23.

Table 23 – NVMe_RJT 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	Link Service 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).

Link Service Reject descriptor: contains a Link Service 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 24.

Table 24 – NVMe_ACC 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) depending on the request being accepted.

8.5 Create Association (CASS)

The Create Association request is used to create an NVMeoFC association between an NVMe host and an NVM subsystem, and an NVMeoFC connection corresponding to the Admin Queue for this NVMeoFC association.

The format of the Create Association request payload is specified in table 25.

Table 25 – 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 descriptor		
257				LSB

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

Create Association descriptor: contains a Create Association descriptor (see 8.2.4) that describes the parameters for the Admin Queue that corresponds to the NVMeoFC association to be created.

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

Table 26 – 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.

Link Service Request Information descriptor: contains a Link Service Request Information descriptor (see 8.2.2) that describes the command for which this is a response.

Association Identifier descriptor: contains an Association Identifier descriptor (see 8.2.8) that identifies the newly created association.

Connection Identifier descriptor: contains a Connection Identifier descriptor (see 8.2.7) that identifies the NVMeoFC connection corresponding to the Admin Queue for the newly created association.

8.6 Create I/O Connection (CIOC)

The Create I/O Connection request is used to create an NVMeoFC connection for an NVMeoFC association.

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

Table 27 – 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 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) that identifies the association for which the I/O connection is to be established.

Create I/O Connection descriptor: contains a Create I/O Connection descriptor (see 8.2.5) that describes the parameters for the I/O Queue that corresponds to the NVMeoFC connection to be created.

The format of the Create I/O Connection accept payload is specified in table 28.

Table 28 – 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.

Link Service Request Information descriptor: contains a Link Service Request Information descriptor (see 8.2.2) that describes the command for which this is a response.

Connection Identifier descriptor: contains a Connection Identifier descriptor (see 8.2.7) that identifies the newly created NVMeoFC connection.

8.7 Disconnect (DISC)

The Disconnect request is used to terminate an NVMeoFC association.

The format of the Disconnect request payload is specified in table 29.

Table 29 – 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		
5				LSB
6	MSB	Disconnect descriptor		
11				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) that identifies the association that corresponds to the requested disconnect.

Disconnect descriptor: contains a Disconnect descriptor (see 8.2.6) that specifies the disconnect action to be taken.

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

Table 30 – 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) that describes the command for which this is a response.

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

9.1 Overview

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

NVMeoFC IUs and their characteristics are specified in table 31 for IUs sent to target NVMe_Ports.

Table 31 – 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 32).

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

NVMeoFC IUs and their characteristics are specified in table 32 for IUs sent to initiator NVMe_Ports.

Table 32 – 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 ERSP Result field set to INVALID FIELD (see table 38). The format of the NVMe_CMND IU is specified in table 33.

Table 33 – 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 NVMeoFC.

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 34.

Table 34 – 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 ERSP Result 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 specific SQ the SQE is to be submitted to. As the NVMe Connection Identifier is unique to an association, it indirectly identifies the NVMe controller in the NVM subsystem.

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 Read and Write 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 in little-endian format (see NVM Express and NVMe over Fabrics).

9.3 NVMe_XFER_RDY IU format

The format of the NVMe_XFER_RDY IU is specified in table 35.

Table 35 – 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) relative to the first byte of the Data Series for the Exchange.

Burst Length: The Burst Length field contains the number of bytes to be transferred in the requested NVMe_DATA IU.

The sum of the value of the Burst Length field and the value of the Relative Offset field shall be less than or equal to the value of the Data Length field and the value in the Burst Length field shall not be zero, otherwise an NVMeoFC data transfer error is detected (see 11.2).

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 NVMeoFC I/O operation is transmitted in the same Exchange that sent the NVMe_CMND IU requesting the transfer.

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

a) if first burst is being used, the first burst NVMe_DATA IU shall be no longer than the First Burst Size field value (see 6.3.3);
b) NVMe_DATA IUs for data for a write operation, 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) the total size of all 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 data for a write operation, 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 detect an NVMeoFC data transfer error (see 11.2).

If an NVMe_XFER_RDY IU is used to request 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 40).

All NVMe_DATA IUs for a write operation, excluding the first burst NVMe_DATA IU if applicable, shall be sent in response to 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 NVMeoFC 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 request transfer of data for a write operation, 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 all of the write data sent in a first burst NVMe_DATA IU.

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 37) shall indicate a value that reflects the amount of data transferred up until the point of the error, and the target NVMe_Port in conjunction with the NVMe controller, shall set the NVMe CQE Status to an appropriate value (see NVM Express) to reflect the error.

9.4.2 NVMe_DATA IUs for read and write operations

The target NVMe_Port shall not request or deliver data outside the buffer length defined by the Data Length field value.

If an SQE requested that data beyond the length specified by the Data Length field in the NVMe_CMND IU be transferred, then the target NVMe_Port in conjunction with the NVM controller (see NVM Express) 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 more data than is specified in the Data Length field. 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 and detect an NVMeoFC data transfer error (see 11.2).

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, if sent, shall be set to the number of bytes transferred, not including first burst retransmission, if applicable.

9.4.3 NVMe_Port transfer byte counting

The initiator NVMe_Port shall maintain a byte count of transferred data for the command. The byte count shall:

a) be set to zero upon transmitting the NVMe_CMND IU;
b) be incremented by the amount of payload in each successfully received NVMe_DATA IU when receiving read data;
c) be incremented by the amount of payload in each successfully transmitted NVMe_DATA IU when transferring data for a write operation; and
d) be decremented by the amount of first burst data transmitted if first burst was transmitted for the command and an NVMe_XFER_RDY IU is received with relative offset set to zero.

If an initiator NVMe_Port receives an NVMe_ERSP IU Transferred Data Length field value that does match its transferred byte count and the NVMe_ERSP IU ERSP Result set to zero (i.e., SUCCESS) 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, then an NVMeoFC data transfer error is detected (see 11.2).

The target NVMe_Port shall maintain a byte count of transferred data for the command. The byte count shall:

a) be set to zero upon receiving the NVMe_CMND IU;
b) be incremented by the amount of payload in each successfully transmitted NVMe_DATA IU when transmitting read data;
c) be incremented by the amount of payload in each successfully received NVMe_DATA IU when receiving data for a write operation; and
d) remain zero if first burst was received and discarded.

An NVMe_RSP IU shall only be sent if the byte count is equal to the Data Length field value specified in the NVMe_CMND IU, and when sending an NVMe_ERSP IU the Transferred Data Length field shall be set to the byte count.

9.4.4 NVMe_DATA IU use of fill bytes (see FC-FS-5)

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

NVMe_RSP IU response rules are specified in 4.8. The format of the NVMe_RSP IU is specified in table 36.

Table 36 – 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

NVMe_RSP IU response rules are specified in 4.9. The format of the NVMe_ERSP IU is specified in table 37.

Table 37 – 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	ERSP Result	Reserved	ERSP IU Length	
1	Response Sequence Number			
2	Transferred Data Length			
3	Reserved			
4	NVM Completion Queue Entry (16 bytes)			
5				
6				
7				

ERSP Result: the ERSP Result field contains an NVMeoFC specific status value specified in table 38.

Table 38 – ERSP Result 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, inclusive of word 0.

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 (see 9.4) 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 in little-endian format (see NVM Express and NVMe over Fabrics) for the NVM command corresponding to the NVMeoFC Exchange.

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 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) that reports NVM subsystems local to the target NVMe_Port by:

a) having an NVM subsystem with a controller that supports the Discovery Service;
b) setting the Discovery Service Supported bit to one in a PRLI LS_ACC;
c) registering FC-4 Features object Discovery Service Supported bit; and
d) responding to the Get Log Page command sent to the Admin Queue of the Discovery Service with an inventory of known NVM subsystems with which an NVMe host may attempt to form an association.

NOTE 2 – 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 NVMeoFC is specified in table 39.

Table 39 – Discovery Log Page for NVMeoFC

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 to 15	Transport Service ID (TRSVCID) (32 bytes)			
16 to 63	Reserved			
64 to 127	NVMe Qualified Name (SUBNQN) (256 bytes)			
128 to 191	Transport Address (TRADDR) (256 bytes)			
192 to 255	Transport Specific Address Subtype (TSAS) (256 bytes)			

Transport Type: shall be set to 02h (i.e., Fibre Channel Transport) (see NVMe over Fabrics).

Address Family: shall be set to 04h (i.e., Fibre Channel address family) (see NVMe over Fabrics).

Subsystem Type: see NVMe over Fabrics.

Transport Requirements: see NVMe over Fabrics.

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

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

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

Transport Service ID: shall be set to the ASCII string “none” (see NVMe over Fabrics).

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

Transport Address: shall be set to “nn-0xWWNN:pn-0xWWPN” where:

a) WWNN is the Node_Name of the target NVMe_Port; and
b) WWPN is the N_Port_Name of the target NVMe_Port.

The WWNN and WWPN are the ACSII values of the sixteen hex digits of the Name_Identifiers.

The Transport Address field is not NULL terminated (see NVM Express) and shall be padded with spaces.

A Transport Address example is "nn-0x20000090FA123456:pn-0x10000090FA123456” followed by 213 spaces.

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

10.2 Transport specific status

The value range B0-BFh (see NVMe over Fabrics) is defined for transport specific errors.

Transport specific status values for NVMeoFC are specified in table 40.

Table 40 – NVMeoFC layer 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);
b) an NVMe_XFER_RDY IU received on an Exchange where the NVMe_CMND IU Flags field had the Read bit set to one;
c) an NVMe_Data IU received on an Exchange where the NVMe_CMND IU Flags field had the Write bit set to one.
d) 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;
e) 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; and
f) an NVMeoFC data transfer error.

A target NVMe_Port shall detect the following:

a) a Sequence error (see FC-FS-5);
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;
c) the length of the NVME_DATA IU is different than the length specified in the Burst Length field; and
d) an NVMeoFC data transfer error.

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).

11.3 Exchange level termination and resource 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. An NVMe I/O Exchange 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 NVMeoFC connection and NVMeoFC association that were 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 shall transmit 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.

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 terminate the designated Exchange and return one of the following responses:

a) if the Nx_Port issuing the ABTS-LS is not currently logged in (i.e., no N_Port Login exists), then the receiving NVMe_Port shall discard the ABTS-LS and transmit a LOGO ELS;
b) if the received ABTS-LS contains an assigned RX_ID field value and a FQXID that is unknown to the receiving NVMe_Port, then 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); 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.

If the RX_ID field is set to FFFFh, then the 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 Additional error recovery by initiator NVMe_Port

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. For example, the NVMe host 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 host 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.

11.3.5 Additional error recovery by target NVMe_Port

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.

11.4 Second-level error recovery

11.4.1 ABTS-LS error recovery

If a response to an ABTS-LS is not received within two times R_A_TOV, then the NVMe_Port may transmit the ABTS-LS 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 NVMeoFC connections and NVMeoFC associations with the corresponding NVMe_Port, shall be terminated at the NVMe_Port.

11.5 Responses to frames before PLOGI or PRLI

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 an explicit PLOGI ELS request, 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 shall terminate 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 an explicit PRLI ELS request 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 shall terminate the Exchange associated with the discarded request.

If an NVMeoFC device receives a frame of category 0001b or 0011b (i.e., solicited data or solicited control) and the NVMeoFC device has not performed successful explicit PLOGI and PRLI with the source of the frame, then the NVMeoFC device shall discard and ignore the content of the frame. If login is not completed, then the NVMeoFC device may transmit a LOGO ELS request to the source of the unexpected frame. If login is completed, but PRLI is not completed, then the NVMeoFC 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 41 – 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
n/a - Not applicable

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 may immediately reuse the Sequence_Qualifier after receiving a BA_ACC to an ABTS-LS.

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. This means multi-byte values are transmitted 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. This means multi-byte values are transmitted 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 the IU payload, the IU will be treated as a raw bytestream and each area is in its native endianness. The Fibre Channel area is big-endian and the NVM Express area is 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 the NVM Express 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 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)								(MSB)						CMND IU Length									(LSB)
1	Reserved																															
2	(MSB)								NVMe Connection Identifier																							
3																																(LSB)
4	(MSB)								Command Sequence Number																							(LSB)
5	(MSB)								Data Length																							(LSB)
6	Opcode (OPC)								PSDT		Reserved				FUSE		(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																														SGL Descriptor Type	Reserved /Zero
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. The CQE follows the format for a Completion

Queue Entry as defined in the NVMe Express specification. As a reminder, all multi-byte fields for the NVM Express area are 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	ERSP Result								Reserved								(MSB)						ERSP IU Length									(LSB)
1	(MSB)								Response Sequence Number																							(LSB)
2	(MSB)								Transferred Data Length																							(LSB)
3	Reserved																															
4	(LSB)								Command Specific																							(MSB)
5	Reserved																															
6	(LSB)				SQ Head Pointer											(MSB)	(LSB)				SQ Identifier											(MSB)
7	(LSB)				Command Identifier											(MSB)	Status Field (LSB)								P	Status Field (MSB)							

Annex B
(informative)
NVMeoFC command IU examples

B.1 Overview

The steps given in the following examples indicate the normal exchange of NVMeoFC 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 data for a write operation may be command data on operations that are not LBA-relative, or LBA 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 data for a write operation 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 data for a write operation;
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 data for a write operation may be command data on operations that are not LBA-relative, or LBA 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 data for a write operation. The data for a write operation starts at offset 0. The length of the data is subject to the values determined 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 data for a write operation is to transferred, the NVMe layer makes one or more requests to transfer the data for a write operation 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 data for a write operation;

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)
NVMeoFC initialization and device discovery

C.1 NVMeoFC 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 NVMeoFC 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., NVMeoFC). This query obtains a list of the Port Identifiers (see FC-GS-8) of devices that support the NVMeoFC 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-oF Connect command via the newly created transport Admin Queue connection. The Connect command is to create the Admin Queue.
      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.
      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.

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., NVMeoFC target function supported), and the Type code field set to 28h (i.e., NVMeoFC). This query obtains a list of the Port Identifiers (see FC-GS-8) of devices that support the NVMeoFC 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 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 Node_Name and N_Port_Name of the target.
        1) interact with the FC Name Server to resolve the Node_Name and N_Port_Name 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:
        1) 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.
    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.
    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 I/O Queue connection. The Connect command is to create the IO Queue.

3) the NVMe layer may perform additional commands on the newly-created IO Queue and connection, such as authentication commands

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.

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 NVMeoFC 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) The NVMe_Port with the highest N_Port_Name sends PLOGI;
2) The initiator NVMe_Port sends 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 Node_Name and N_Port_Name of the target NVMe_Port.
      1) the Node_Name and N_Port_Name 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 terminate 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 (see C.1.1) 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
CSN	Command Sequence Number
L	Last Sequence

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

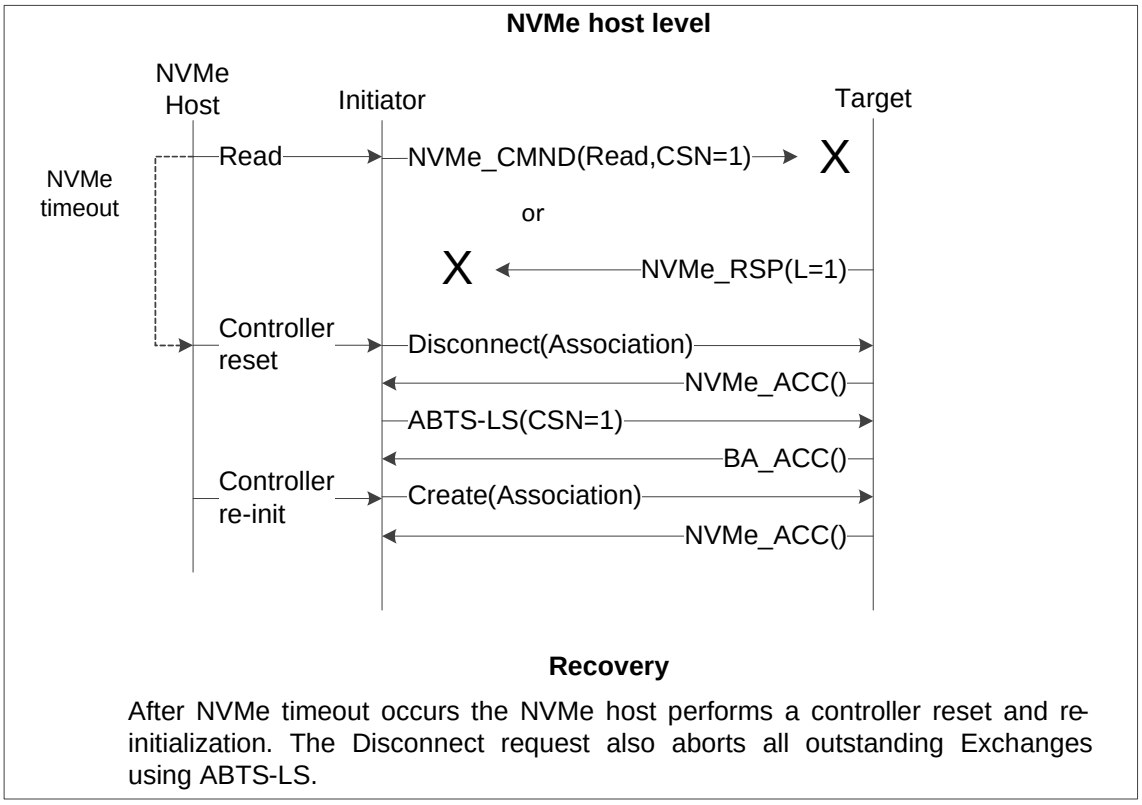

Figure D.1 - NVMe_CMND lost or NVMe_RSP lost with controller reset

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

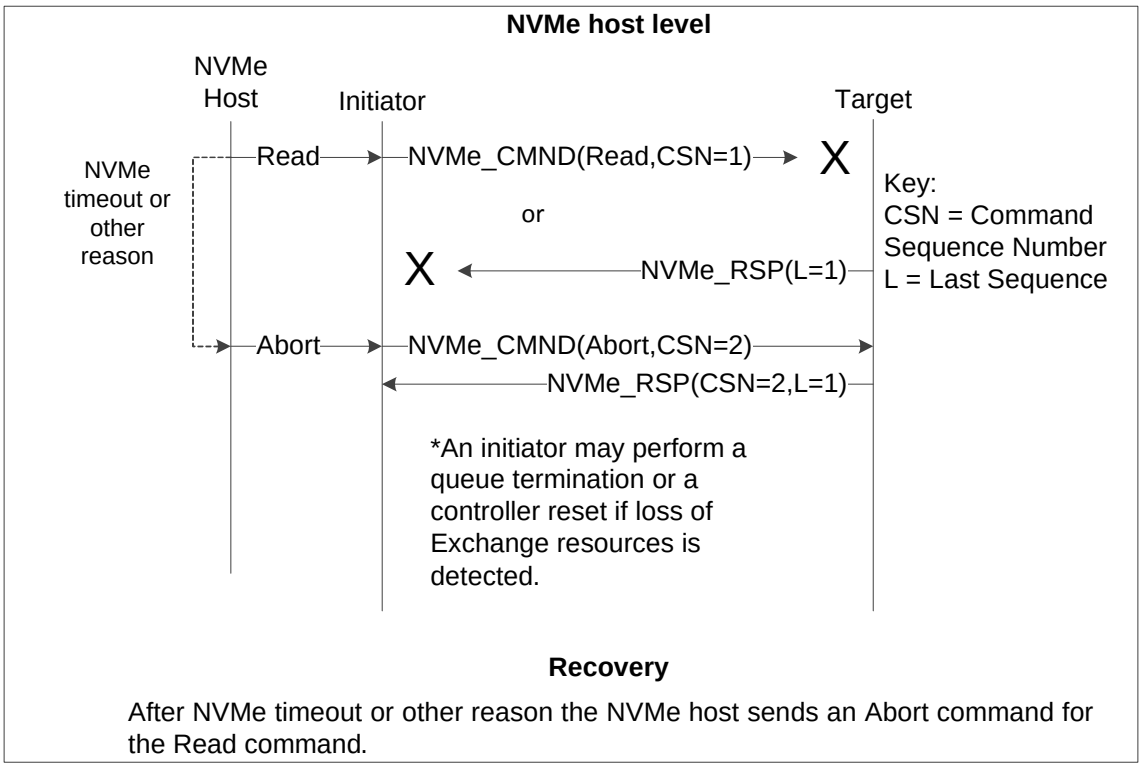

Figure D.2 - NVMe_CMND lost or NVMe_RSP lost with NVM Abort

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

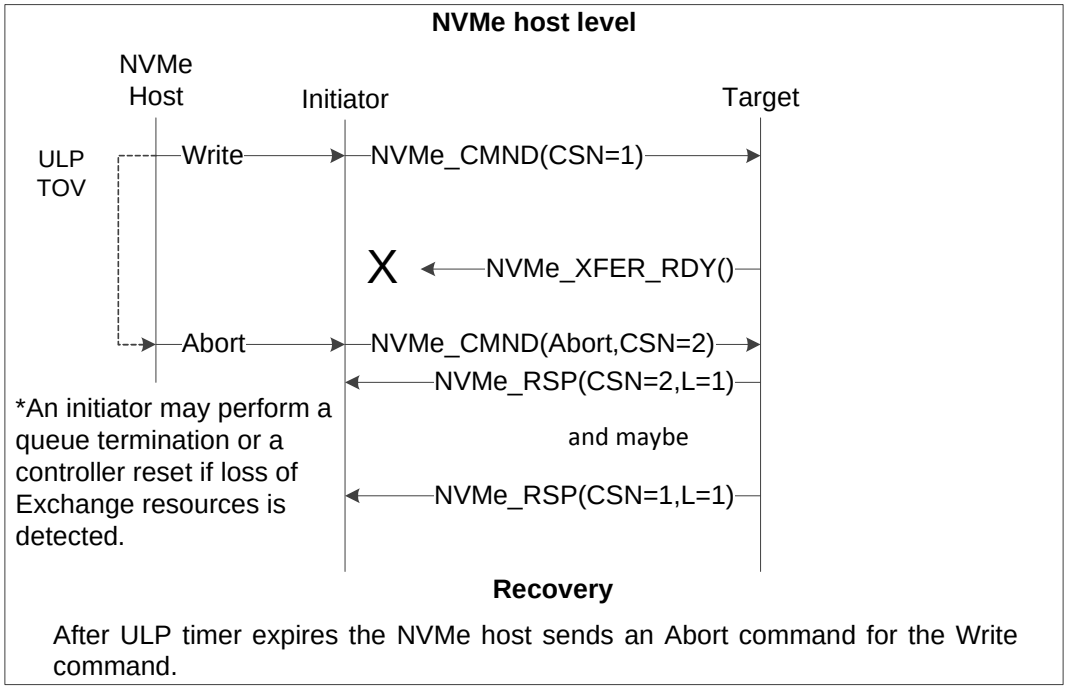

Figure D.3 - NVMe_XFER_RDY lost with NVM Abort