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

INCITS 556-201x
T11/Project 556-D/Rev 1.04

FIBRE CHANNEL

NON-VOLATILE MEMORY EXPRESS — 2

(FC-NVMe-2)

REV 1.04

INCITS working draft proposed
American National Standard
for Information Technology

February 6, 2019

Secretariat: Information Technology Industry Council

POINTS OF CONTACT:

Steven Wilson (T11 Chair)
Broadcom Inc
1320 Ridder Park Drive
San Jose, CA 95131
Voice: 408-433-8000
steve.wilson@broadcom.com

Craig W. Carlson (T11 Vice Chair)
Marvell Semiconductor
12900 Whitewater Drive
Minnetonka, MN 55343
Voice: 952-687-2431
craig.carlson@marvell.com

Craig W. Carlson (T11.3 Chair)
Marvell Semiconductor
12900 Whitewater Drive
Minnetonka, MN 55343
Voice: 952-687-2431
craig.carlson@marvell.com

Craig W. Carlson (FC-NVMe-2 Chair)
Marvell Semiconductor
12900 Whitewater Drive
Minnetonka, MN 55343
Voice: 952-687-2431
craig.carlson@marvell.com

David Peterson (FC-NVMe-2 Editor)
Broadcom Inc
1230 Northland Drive
Mendota Heights, MN 55120
Voice: 408-433-8000
david.peterson@broadcom.com

Revision History

Rev 1.04
Updated SLER informative annex diagrams.

Rev 1.03
(2018-00155-v005) - SLER informative annex diagrams
(2018-00328-v001) - Lost Read Data Enhancement
(2018-00262-v001) - NVMe_CMND IU Extension for PI

Rev 1.02
(2018-00221-v002) - Proposed modifications to FC-NVMe-2 due to NVMe over Fabrics TP 8005
(2018-00261-v001) - Lost command Exchange handling

Rev 1.01
(2018-00103-v003) - FC-NVMe-2 Sequence level error recovery additions
(2018-00047-v007) - FC-NVMe-2 Sequence Level Error Recovery
(2018-00129-v002) - FC-NVMe(-2) and NVMe Keep Alives (slide 7)

Rev 1.00
(2017-00419-v004) - Revisions to FC-NVME-2 for change in delete association actions by initiator and target NVMe_Ports.
(2018-00040-v001) - Admin Command determinism (slide 4)

Rev 0.00
- initial draft standard based on FC-NVMe standard

BSR INCITS 556-201x

American National Standard
for Information Technology

Fibre Channel - NVMe — 2 (FC-NVMe-2)

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.

Published by

American National Standards Institute
25 West 43rd Street, 4th floor New York, NY 10036

Foreword (This Foreword is not part of American National Standard INCITS 556-201x.)

This standard defines a Fibre Channel mapping layer (FC-4) that uses the services defined by INCITS Project 562-D, Fibre Channel Framing and Signaling Interface - 6(FC-FS-6) 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 2017-2019. The standards approval process started in 2019. 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)

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-2 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 — 2 (FC-NVMe-2)

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.

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

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

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

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

INCITS 544:2016, *Fibre Channel - Single Byte Command Code Sets - 6 (FC-SB-6)*

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

INCITS 481-2011, *Fibre Channel Protocol for SCSI - 4 (FCP-4*)

NVM Express revision 1.3 - May 1, 2017

NVMe over Fabrics revision 1.0 - June 5, 2016

NVMe TP4008, *Transport SGL*

3 Definitions, abbreviations, symbols, keywords, and conventions

3.1 Definitions

3.1.1 Association Identifier
value that uniquely identifies an NVMeoFC association (see 4.3)

3.1.2 Connection Identifier
value that uniquely identifies an NVMeoFC connection (see 4.4)

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 without waiting for an NVMe_XFER_RDY 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 a Submission Queue (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-6) 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-4	Fibre Channel - Link Services - 4
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-2, 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-6 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-6 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.

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-6.

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-6). Link control is performed by standard FC-FS-6 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 29 and table 30.

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-6 implementation. The architectural limit for this value is 65 535.

NVMeoFC protocol layers are shown in figure 1.

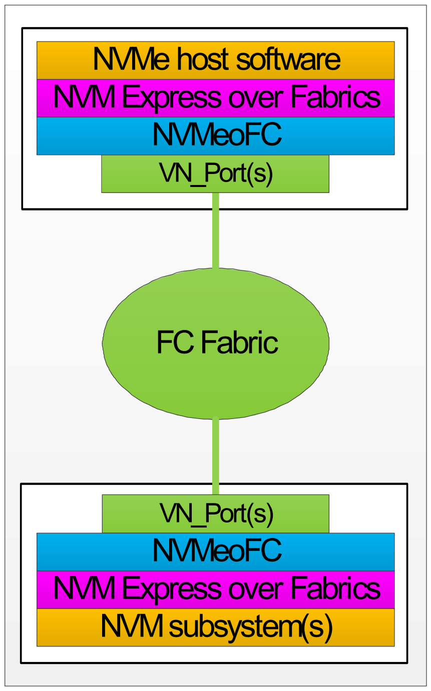


Figure 1 – NVMeoFC protocol layers

The NVMeoFC 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.

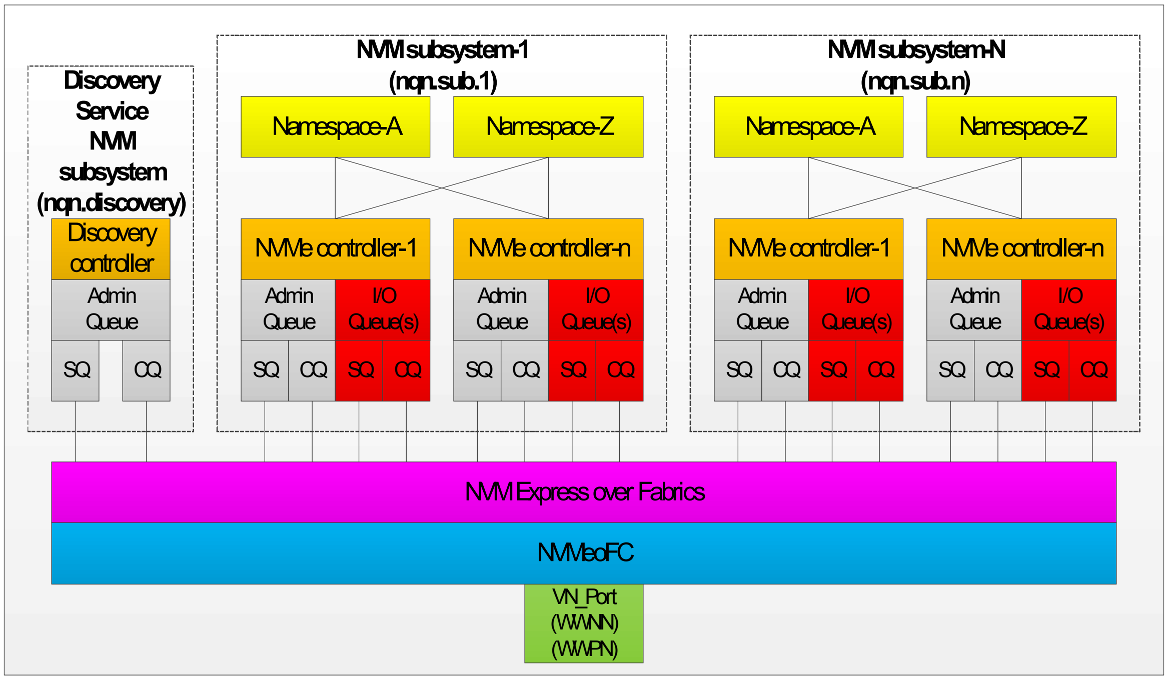


Figure 2 – NVMeoFC target device functional model

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 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 41) 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

4.3.1 NVMeoFC association overview

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 8.3.3). 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 8.3.3) 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 8.3.1) to the initiator NVMe_Port with the reason code and reason code explanation set to an appropriate value.

An NVMe over Fabrics association is established when the NVMe-oF Connect command (see NVMe over Fabrics) to create the Admin Queue is issued on the NVMeoFC association's Admin Queue connection. Refer to NVMe over Fabrics for additional requirements and behaviors of NVMe over Fabrics associations and Admin Queue creation.

An active NVMeoFC association shall be terminated if:

a) any NVMeoFC connection (see 4.4) for the NVMeoFC association is terminated;
b) a Disconnect NVMe_LS request (see 8.3.5) is received with an Association Identifier descriptor containing the Association Identifier of the NVMeoFC association;
c) one of the clearing effects (see 4.15) 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.

An initiator NVMe_Port or a target NVMe_Port may terminate an NVMeoFC association. While an active NVMeoFC association is being terminated, the NVMe_Port shall:

a) not transmit NVMe_CMND IUs or NVMe_LS requests other than a Disconnect NVMe_LS, that contain the Association Identifier or Connection Identifiers that correspond to the NVMeoFC association being terminated;
b) discard any received NVMe_CMND IUs or NVMe_LS requests other than a Disconnect NVMe_LS, that contain the Association Identifier or Connection Identifiers that correspond to the NVMeoFC association being terminated;
c) if the NVMe_Port is an initiator NVMe_Port, then perform the initiator NVMe_Port NVMeoFC association termination process (see 4.3.2); and
d) if the NVMe_Port is a target NVMe_Port, then perform the target NVMe_Port NVMeoFC association termination process (see 4.3.4).

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

4.3.2 Initiator NVMe_Port NVMeoFC association termination process

The initiator NVMe_Port NVMeoFC association termination process is performed by an initiator NVMe_Port to terminate the NVMeoFC association and associated NVMeoFC connections, and recover all outstanding exchange resources that are associated with the NVMeoFC association to be terminated.

An initiator NVMe_Port shall perform the following steps to terminate an NVMeoFC association:

1) transmit an ABTS-LS (see 11.3) for all open Exchanges for the NVMeoFC association being terminated, except for a Disconnect NVMe_LS Exchange;
2) transmit a Disconnect NVMe_LS request to the corresponding target NVMe_Port, unless there is no valid login with the associated NVMe_Port. The Disconnect NVMe_LS request, if transmitted, shall contain an Association Identifier descriptor with the Association Identifier of the NVMeoFC association being terminated; and
3) await recovery of all open Exchange resources as described in 4.3.3 for the NVMeoFC association being terminated.

If the initiator NVMe_Port has received or receives a Disconnect NVMe_LS request for the NVMeoFC association being terminated, then the initiator NVMe_Port shall not transmit the Disconnect NVMe_LS response until the initiator NVMe_Port has completed steps 1 and 2 of the initiator NVMe_Port NVMeoFC association termination process for the association to be terminated.

If the Disconnect NVMe_LS response to the Disconnect NVMe_LS request, if transmitted as described in step 2 in this subclause, is received after the NVMeoFC association is terminated, then the response shall be ignored by the initiator NVMe_Port.

If a response to the Disconnect NVMe_LS request, if transmitted as described in step 2 in this subclause, is not received within two times R_A_TOV, then the initiator NVMe_Port shall:

a) send an ABTS-LS to recover the Exchange resources for the Disconnect NVMe_LS request; and
b) if the Exchange resources for all open Exchanges associated with the NVMeoFC association to be terminated have not been recovered, the initiator NVMe_Port shall transmit a second Disconnect NVMe_LS request containing the same request payload in a separate Exchange, or transmit a LOGO ELS to the corresponding target NVMe_Port. If a response was not received for the second Disconnect NVMe_LS request, if transmitted, after two times R_A_TOV, then the initiator NVMe_Port shall transmit a LOGO ELS to the corresponding target NVMe_Port.

4.3.3 Initiator NVMe_Port Exchange recovery

For an NVMeoFC association being terminated, an initiator NVMe_Port shall consider an open Exchange resource recovered:

a) upon the reception of an ABTS-LS from the corresponding target NVMe_Port for the Exchange;
b) upon the reception of a BA_ACC or BA_RJT to the ABTS-LS issued for the Exchange as described in step 1 in 4.3.2;
c) after a delay of R_A_TOV after the reception of a Disconnect NVMe_LS request that contains the Association Identifier descriptor with the Association Identifier set to the identifier of the NVMeoFC association being terminated;
d) after a delay of R_A_TOV after the reception of the Disconnect NVMe_LS response to the Disconnect NVMe_LS request transmitted as described in step 2 in 4.3.2; or
e) upon a logout or process logout with the target NVMe_Port.

4.3.4 Target NVMe_Port NVMeoFC association termination process

The target NVMe_Port NVMeoFC association termination process is performed by a target NVMe_Port to terminate the NVMeoFC association and associated NVMeoFC connections, recover all outstanding exchange resources that are associated with the NVMeoFC association to be terminated, and recover the Association Identifier and Connection Identifiers corresponding to the NVMeoFC association to be terminated.

A target NVMe_Port shall perform the following steps to terminate an NVMeoFC association:

1) transmit an ABTS-LS (see 11.3) for each open Exchange for the NVMeoFC association being terminated, except for a Disconnect NVMe_LS Exchange;
2) transmit a Disconnect NVMe_LS request to the corresponding initiator NVMe_Port unless there is no valid login with the associated NVMe_Port. The Disconnect NVMe_LS request, if transmitted, shall contain an Association Identifier descriptor with the Association Identifier set to the identifier of the NVMeoFC association being terminated;
3) wait for:
   A) reception of a Disconnect NVMe_LS request from the intiator NVMe_Port that contains the Association Identifier descriptor with the Association Identifier set to the identifier of the NVMeoFC association being terminated;
   B) reception of the Disconnect NVMe_LS response to the Disconnect NVMe_LS request, if transmitted, as described in step 2 in this subclause. If that response is an NVMe_RJT and the Link Service Reject descriptor has the reason code set to a value other than 40h (i.e., Invalid Association Identifier), then the target NVMe_Port shall wait for the condition described in step 3)A) in this subclause or for the condition described in step 3)C) in this subclause. If that response is an NVMe_ACC or that response is an NVMe_RJT and the Link Service Reject descriptor is set to a reason code of 40h (i.e, Invalid Association Identifier), the target NVMe_Port may proceed with step 4 in this subclause; or
   C) a delay of four times R_A_TOV. After that delay, the target NVMe_Port shall transmit a LOGO ELS to the corresponding initiator NVMe_Port.
4) await recovery of all open Exchange resources as described in 4.3.5 for the NVMeoFC association being terminated; and
5) make the Association Identifier and Connection Identifiers corresponding to the terminating NVMeoFC association available for re-use:
   A) after a R_A_TOV delay; or
   B) upon a logout or Process Logout with the initiator NVMe_Port.

If the target NVMe_Port has received or receives a Disconnect NVMe_LS request for the NVMeoFC association being terminated, the target NVMe_Port shall not transmit the Disconnect NVMe_LS response until the target NVMe_Port has completed steps 1 and 2 of the target NVMe_Port NVMeoFC association termination process for the association to be terminated.

If the Disconnect NVMe_LS response to the Disconnect NVMe_LS request, if transmitted as described in step 2 in this subclause, is received after the NVMeoFC association is terminated, then the response shall be ignored by the target NVMe_Port.

If the Disconnect NVMe_LS response to the Disconnect NVMe_LS request, if transmitted as described in step 2 in this subclause, is not received within two times R_A_TOV, then the target NVMe_Port shall:

a) send an ABTS-LS to recover the Exchange resources for the Disconnect NVMe_LS request; and
b) if the Exchange resources for all open Exchanges associated with the NVMeoFC association to be terminated have not been recovered, then the target NVMe_Port shall transmit a second Disconnect NVMe_LS request containing the same request payload in a separate Exchange, or transmit a LOGO ELS to the corresponding initiator NVMe_Port. If a response was not received for the second Disconnect NVMe_LS request, if transmitted, after two times R_A_TOV, then the target NVMe_Port shall transmit a LOGO ELS to the corresponding initiator NVMe_Port.

4.3.5 Target NVMe_Port Exchange recovery

For an NVMeoFC association being terminated, a target NVMe_Port shall consider an open Exchange resource recovered upon:

a) the reception of an ABTS-LS from the corresponding initiator NVMe_Port for the Exchange;
b) the reception of a BA_ACC or BA_RJT to the ABTS-LS issued for the Exchange as described in step 1 in 4.3.4;
c) the reception of an NVMe_ACC response to the Disconnect NVMe_LS request, if transmitted, as described in step 2 in 4.3.4;
d) the reception of an NVMe_RJT response to the Disconnect NVMe_LS request, if transmitted as described in step 2 in 4.3.4, and that response has the Link Service Reject descriptor set to a reason code 40h (i.e., Invalid Association Identifier).
e) the reception of a Disconnect NVMe_LS request that contains the Association Identifier descriptor with the Association Identifier set to the identifier of the NVMeoFC association being terminated; or
f) a logout or Process Logout with the initiator NVMe_Port.

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 corresponds to either 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 8.3.3). 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 8.3.3) contains a Connection Identifier 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 Identifier, initiator NVMe_Port N_Port_ID, target NVMe_Port N_Port_ID> shall be unique.

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 8.3.4) 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 8.3.4) to the initiator NVMe_Port. The Create I/O Connection NVMe_ACC payload contains a Connection Identifier 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 Identifier, 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 Identifier) 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 and reason code explanation set to an appropriate value.

The first NVMeoFC I/O operation issued on an NVMeoFC connection contains an NVMe-oF Connect command (see NVMe over Fabrics) for the corresponding NVMe controller queue.

The values of the Host Identifier, Host NVMe Qualified Name, and NVM Subsystem NVMe Qualified Name fields in any NVMe-oF Connect command shall match the values transmitted in the Create Association NVMe_LS that created the NVMeoFC association for the NVMeoFC connection. If the queue is the Admin Queue (i.e., the Queue ID field is set to zero), then the values of the Submission Queue Size and Controller ID in the NVMe-oF Connect command shall match the values transmitted in the Create Association NVMe_LS that created the NVMeoFC connection corresponding to the Admin Queue. If the queue is an I/O Queue (i.e., the Queue ID field is non-zero), then the values of the Submission Queue Size and Queue ID fields in the NVMe-oF Connect command shall match the values transmitted in the Create I/O Connection NVMe_LS that created the NVMeoFC connection, and the value of the Controller ID field shall match the Controller ID value transmitted in the Connect Response returned by the target NVMe_Port for the NVMe-oF Connect command issued on the NVMe-oFC connection when the Admin Queue was created. If any of the values do not match, the target NVMe_Port shall transmit an NVMe_ERSP IU for the NVMe I/O operation for the NVMe-oF Connect command with ERSP Result field set to 03h (i.e., ILLEGAL CONNECT PARAMETERS), the Transferred Data Length field set to zero, and the target NVMe_Port shall terminate the NVMeoFC connection. See NVMe over Fabrics for additional requirements and behaviors of I/O Queue creation.

An active NVMeoFC connection is terminated if:

a) the NVMeoFC association for the NVMeoFC connection is terminated;
b) the NVMe-oF Connect command has field values inconsistent with the NVMe_LS commands used to create the related NVMeoFC association or NVMeoFC connection as described in this subclause;
c) an error is detected on an NVMeoFC I/O operation (see 11.2) associated with the NVMeoFC connection; or
d) the NVMe host or NVM subsystem detects a condition which causes it to terminate the corresponding NVMe-oF connection.

The initiator NVMe_Port or the target NVMe_Port may terminate an NVMeoFC connection.

The termination of an NVMeoFC connection terminates the associated NVMe Queue. The termination of the Queue 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 Identifier, 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 Identifier) 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 Identifier, 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-6) 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 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 4), then the initiator NVMe_Port may choose to perform write operations by sending a first NVMe_DATA IU (see 9.4) no longer than the First Burst Size field value (see 6.3.3), 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 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) if the NVMe Host and the NVMe Controller have not negotiated to disable Submission Queue flow control (see Submission Queue Flow Control Negotiation in the NVMe over Fabrics specification) for the Submission Queue that corresponds to the NVMeoFC connection that would be used to send the NVMe_RSP IU, then;
   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; and
   B) an NVMe_ERSP IU shall be sent by the target NVMe_Port if the SQ is 90% or more full (see NVM Express);
b) 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);
c) 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);
d) 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
e) an NVMe_ERSP IU may be sent by the target NVMe_Port for any other reason.

NOTE 1 – The result of the Submission Queue Flow Control Negotiation process is communicated between the NVM Express over Fabrics layer and the NVMeoFC layer (see figure 1) by means outside the scope of this standard.

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:
   A) zero, if no NVMe_ERSP IU has been delivered to the NVMe host on the same connection; and
   B) the SQHD value of the CQE in the most recent NVMe_ERSP IU that was delivered to the NVMe host on the same connection, if an NVMe_ERSP IU has been delivered to the NVMe host on the same connection;
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 9.4.4), 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), then an NVMeoFC transport error has occurred (see 11.2).

If a command is completed by an NVMe_ERSP IU with a Transferred Data Length field value that is not equal to the initiator NVMe_Port's transfer byte count, then an NVMeoFC data transfer error has occurred (see 11.2).

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, then a target NVMe_Port may request an NVMe_CONF IU by setting the Last_Sequence bit to zero (see FC-FS-6) 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 with 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.

If an initiator NVMe_Port and a target NVMe_Port indicate support of confirmed completion and do not indicate support of Sequence level error recovery (see table 4), then an NVMe_CONF IU should not be requested 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.

4.11 Data transfer

4.11.1 Overview

NVMeoFC rules are specified in 4.11.2 for the following types of NVMe-oF data transfers:

a) SGL data for write operations; and
b) SGL data for read operations.

4.11.2 SGL data

4.11.2.1 Overview

NVMe over Fabrics defines a mechanism for transmitting SGLs across a Fabric. Fibre Channel does not send SGLs across the Fabric. SGLs shall be converted to data sent within a Data Series for transmission across a Fibre Channel Fabric.

4.11.2.2 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 4.11.2.3).

An SGL example is shown in figure 3.

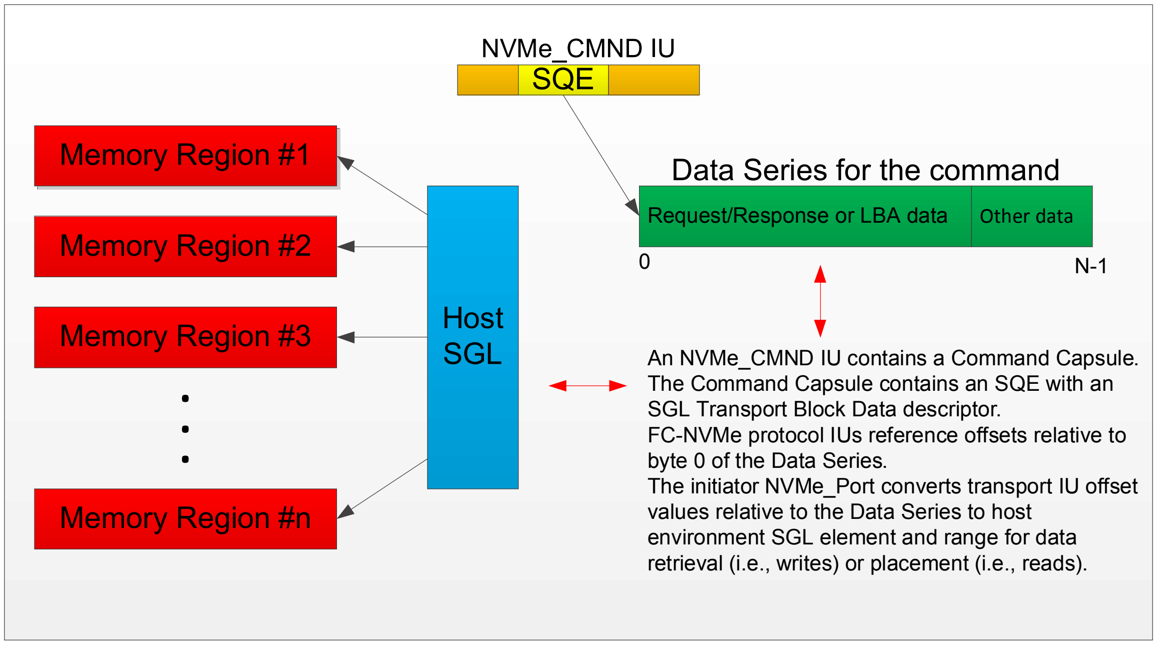

NVMe_CMND IU
SQE
Data Series for the command
Request/Response or LBA data
Other data
0
N-1
Memory Region #1
Memory Region #2
Memory Region #3
Memory Region #n
Host SGL
An NVMe_CMND IU contains a Command Capsule.
The Command Capsule contains an SQE with an SGL Transport Block Data descriptor.
FC-NVMe protocol IUs reference offsets relative to byte 0 of the Data Series.
The initiator NVMe_Port converts transport IU offset values relative to the Data Series to host environment SGL element and range for data retrieval (i.e., writes) or placement (i.e., reads).

Figure 3 – SGL example

4.11.2.3 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 5h (i.e., Transport SGL Data Block descriptor);
b) the SGL Descriptor Sub Type field shall be set to Ah (i.e., NVMe Transport Specific);
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 Sequence level error recovery capability

Sequence level error recovery (SLER) error detection and IU retransmission algorithms are defined in clause 11.

The SLER 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 support for SLER. If an initiator NVMe_Port and a target NVMe_Port both indicate support of SLER (see table 4), then the SLER capability is successfully negotiated. In this case, Sequence level error recovery and SLER qualification (see 4.13) are supported.

4.13 SLER qualification

SLER qualification provides an additional mechanism for relating commands that are being retried to the requests that are sensing the requirement for recovery (i.e., the FLUSH BLS request).

For example, it is possible that initiator NVMe_Ports may re-use OX_ID field values rapidly enough to create an ambiguous situation where the status being preserved in the target NVMe_Port for possible retransmission and the new command being presented to the target NVMe_Port may have the same

OX_ID field values. When recovery of a transmission failure for the new command is attempted, the target NVMe_Port instead indicates that the recovery is related to the previous command's status and the initiator NVMe_Port is provided status for the completed command. That information is mistakenly interpreted as status for the failed command.

NVMe_Ports that agree to perform Sequence level error recovery shall support SLER qualification. If the initiator NVMe_Port and target NVMe_Port agree to support Sequence level error recovery, then a SLER qualifier shall be provided in the Parameter field of each NVMe_CMND IU frame. The Link Services and IUs associated with retransmission of IUs (i.e., FLUSH BLS and NVMe_SR IU) each contain the same SLER qualifier. If the initiator NVMe_Port and target NVMe_Port do not agree to support Sequence level error recovery, then the Parameter field shall be set to zero for NVMe_CMND IU frames.

The value of the SLER qualifier is outside the scope of this standard.

4.14 NVMeoFC capabilities

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

a) initiator NVMe_Port;
b) target NVMe_Port;
c) Discovery Service;
d) Sequence level error recovery;
e) confirmed completion;
f) first burst; and
g) PI control.

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

4.15 Port Login and Port Logout

The N_Port Login (PLOGI) ELS (see FC-LS-4) is used to establish the Fibre Channel operating parameters between any two Fibre Channel ports, including NVMe_Ports. Implicit login functions are not allowed.

When performing N_Port login, support for the following features shall be indicated:

a) Class 3;
b) continuously increasing relative offset; and
c) relative offset by category for the solicited data information category.

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.

4.16 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.

4.17 Link management

FC-FS-6 allows management protocols above the FC-FS-6 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-6 and FC-LS-4).

4.18 NVMeoFC addressing and Exchange identification

The address of each NVMe_Port is defined by its address identifier as described in FC-FS-6.

Each NVMeoFC association is identified by the Association Identifier created by a successful Create Association NVMe_LS request (see 8.3.3). 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 8.3.4). 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.

4.19 Use of Worldwide_Names

As specified in FC-FS-6, 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-6. 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 2 – 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).

4.20 PI control capability

PI control provides additional fields in the NVMe_CMND IU that may be used to support NVM Express end-to-end data protection (see 9.2).

The PI Control Word Present bit 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 support for PI control. If an initiator NVMe_Port and a target NVMe_Port both indicate support of PI control (see table 4), then the PI control capability is successfully negotiated. In this case, if the PI Control Word Present bit is set to one in the NVMe_CMND IU, then the DPS field, the LBADS field, and the MS field in the NVMe_CMND IU contain values that may be used to support NVM Express end-to-end data protection.

5 FC-FS-6 Frame_Header

The format of the FC-FS-6 Frame_Header is specified in table 1.

Table 1 – FC-FS-6 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-6 Frame_Header are specified in FC-FS-6. The following explanations of the fields provide information about the use of those fields to implement NVMeoFC I/O operations or FC-4 Link Services.

R_CTL: The R_CTL field is subdivided into a ROUTING field and an INFORMATION field (see FC-FS-6). For NVMeoFC I/O operations the ROUTING field shall be set to 0h (i.e., Device_Data) and the INFORMATION field shall be set to a value specified in table 29 and table 30.

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-6 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-6) for all frames of NVMeoFC Exchanges using IUs specified in table 29 and table 30; or
b) 28h (i.e., NVMe over Fibre Channel) (see FC-FS-6) for 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. The Parameter field shall contain a value of zero unless a SLER qualifier is specified.

6 NVMeoFC Link Services

6.1 Overview of Link Service requirements

The NVMeoFC link-level protocol includes the Basic Link Services (see FC-FS-6) 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 effects specified in 11.6.3.

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.

Clearing effects of Process Login are specified in 11.6.3.

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 2.

Table 2 – 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-6).

TYPE Code Extension: Shall be set to 00h.

Common Information: The format of the Common Information field is specified in table 3.

Table 3 – 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 4.

Table 4 – Service Parameter Information field format - request and accept

Bit	Description
31 to 10	Reserved
9	PI Control Supported: If set to one, then PI control is supported (see 4.20). If set to zero, then PI control is not supported.
8	SLER Supported: If set to one, then Sequence level error recovery (see 4.12) and SLER qualification (see 4.13) are supported. If the SLER Supported bit is set to zero, then Sequence level error recovery and SLER qualification are not supported.

Table 4 – Service Parameter Information field format - request and accept

Bit	Description
7	Confirmed Completion Supported: If set to one, then confirmed completion is supported (see 4.10). If set to zero, then confirmed completion is not supported.
6	Reserved
5	Initiator Function: If the Initiator Function bit is set to one, then the Originator or Responder is indicating it has the capability of operating as an initiator NVMe_Port. If the Initiator Function bit is set to zero, then the Originator or Responder does not have the capability of operating as an initiator NVMe_Port.
4	Target Function: If the Target Function bit is set to one, then the Originator or Responder is indicating that it has the capability of operating as a target NVMe_Port. If the Target Function bit is set to zero, then the Originator or Responder does not have the capability of operating as a target NVMe_Port.
3	Discovery Service: If the Discovery Service bit is set to one, then the Originator or the 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. The Discovery Service bit shall only be set to one if the Target Function bit is set to one.
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 5.

Table 5 – 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-6).

TYPE Code Extension: Shall be set to 00h.

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

Table 6 – 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 4.

First Burst Size: If first burst is not supported (see table 4), 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 and first burst is supported, 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.

Clearing effects of Process Logout are specified in 11.6.3.

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 7.

Table 7 – 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-6).

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 8.

Table 8 – 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-6).

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-6).

An NVMe_Port shall register the NVMe over Fibre Channel TYPE (28h) with the Name Server using the RFT_ID request CT_IU, indicating support for all frame types supported by this standard (see clause 5).

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 9.

Table 9 – 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, and an ABTS-LS shall be sent to terminate the Exchange.

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 Identifier) and the reason code explanation set to 00h (i.e., No additional explanation).

The NVMe_LS requests and responses are specified in table 10.

Table 10 – NVMe_LS requests and responses

Value (Bits 31-24)	Request/Response	Description	Abbr.	Reference
01h	Response	NVMe_LS reject	NVMe_RJT	8.3.1
02h	Response	NVMe_LS accept	NVMe_ACC	8.3.2
03h	Request	Create Association	CASS	8.3.3
04h	Request	Create I/O Connection	CIOC	8.3.4
05h	Request	Disconnect	DISC	8.3.5
All others		Reserved		

8.2 NVMe Link Service descriptors

8.2.1 Overview

The NVMe_LS descriptors are specified in table 11.

Table 11 – 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 12.

Table 12 – 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 13.

Table 13 – 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 14.

Table 14 – 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 Identifier	The Association Identifier specified in the NVMe_LS request is not a valid ID.
41h	Invalid Connection Identifier	The Connection Identifier specified in the NVMe_LS request is not a valid ID.
42h	Invalid Parameters	The recipient of the NVMe_LS request has rejected the request due to an invalid payload parameter value. See the reason code explanation field for additional details.
43h	Insufficient Resources	The recipient of the NVMe_LS request has rejected the request due to a lack of resources to fulfill the request.
FFh	Vendor specific	The vendor specific error bits in word 2, bits 7:0 (see table 13) may be used by vendors to specify additional reason codes.
All others	Reserved	

reason code explanation: contains a value specified in table 15.

Table 15 – NVMe_LS reason code explanations

Value	Description	Meaning
00h	No additional explanation	-
17h	Invalid OX_ID-RX_ID combination	-
2Ah	Unable to supply requested data	-
2Dh	Invalid payload length	-

Table 15 – NVMe_LS reason code explanations (Continued)

Value	Description	Meaning
40h	Invalid NVMe_ERSP Ratio	The value of the NVMe_ERSP ratio is equal to or greater than the Submission Queue Size.
41h	Invalid Controller ID	The NVM subsystem does not support the indicated Controller ID, the Controller ID is in use, or the Controller ID is invalid for the NVMe-oF association.
42h	Invalid Queue ID	The number of I/O queues has not been configured on the NVM controller, the NVM controller has not been enabled, the Queue ID exceeds the configured number of I/O queues on the NVM controller, or the Queue ID is already active on the NVM controller.
43h	Invalid Submission Queue Size	The value is 0 or exceeds the maximum submission queue size for the queue type (Admin queue or I/O queue).
44h	Invalid HostID	The value is zero or the NVM subsystem has rejected the Host ID.
45h	Invalid HOSTNQN	The value has a syntax error or the NVM subsystem has rejected the host NQN.
46h	Invalid SUBNQN	The value has a syntax error or the subsystem NQN does not exist.
All others	Reserved	

8.2.4 Create Association descriptor

The format of the Create Association descriptor is specified in table 16.

Table 16 – 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 .. 11	Reserved			
12	Controller ID (CNTLID)		Submission Queue Size (SQSIZE)	
13	Reserved			
14 .. 17	Host Identifier (HOSTID)			
18 .. 81	Host NVMe Qualified Name (HOSTNQN)			
82 .. 145	NVM Subsystem NVMe Qualified Name (SUBNQN)			
146 .. 253	Reserved			

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 (i.e., 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-oF Connect command (see NVMe over Fabrics).

Submission Queue Size: contains a value one less than the size of the Admin Submission Queue to be created as specified in the NVMe-oF Connect command (see NVMe over Fabrics).

Host Identifier: contains the Host Identifier as specified in the NVMe-oF Connect command (see NVMe over Fabrics).

Host NVMe Qualified Name: contains the Host NVMe Qualified Name as specified in the NVMe-oF Connect command (see NVMe over Fabrics).

NVM Subsystem NVMe Qualified Name: contains the NVM Subsystem NVMe Qualified Name as specified in the NVMe-oF Connect command (see NVMe over Fabrics).

8.2.5 Create I/O Connection descriptor

The format of the Create I/O Connection descriptor is specified in table 17.

Table 17 – 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 11	Reserved			
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.

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

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 a value one less than the size of the I/O Submission Queue created as specified in the NVMe-oF Connect command (see NVMe over Fabrics).

8.2.6 Disconnect descriptor

The format of the Disconnect descriptor is specified in table 18.

Table 18 – 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 19.

Table 19 – 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 NVMeoFC connection.

8.2.8 Association Identifier descriptor

The format of the Association Identifier descriptor is specified in table 20.

Table 20 – 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.

8.3 NVMeoFC Link Service requests and responses

8.3.1 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 21.

Table 21 – 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)

8.3.2 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 22.

Table 22 – 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 11) depending on the request being accepted.

8.3.3 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 23.

Table 23 – 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		
255				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 24.

Table 24 – 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.

8.3.4 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 25.

Table 25 – 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 26.

Table 26 – 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.

8.3.5 Disconnect (DISC)

The Disconnect request is used to terminate an NVMeoFC association.

The format of the Disconnect request payload is specified in table 27.

Table 27 – 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).

The format of the Disconnect accept payload is specified in table 28.

Table 28 – 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 29 for IUs sent to target NVMe_Ports.

Table 29 – 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
T5	Sequence Retransmission request	09h	NVMe_SR	M	T	O

Notes:
T2 is only permitted if first burst is used on the command.
T4 is only permitted in response to an I4 or I6 frame (see table 30).

Key:

IU	Information Unit identifier	
Content	Contents (payload) of data block	
F/M/L	First/Middle/Last Sequence of Exchange (FC-FS-6)	
	F	First
	M	Middle
	L	Last
SI	Sequence Initiative: Held or Transferred (FC-FS-6)	
	H	Held
	T	Transferred
M/O	Mandatory/Optional Sequence	
	M	Mandatory
	O	Optional

NVMeoFC IUs and their characteristics are specified in table 30 for IUs sent to initiator NVMe_Ports.

Table 30 – 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
I7	Sequence Retransmission response	0Ah	NVMe_SR_RSP	M	H	O

Notes:
a I4 or I6 is requested by not setting First/Middle/Last Sequence of Exchange (see FC-FS-6) 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-6)	
	F	First
	M	Middle
	L	Last
SI	Sequence Initiative: Held or Transferred (FC-FS-6)	
	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 37). The format of the NVMe_CMND IU is specified in table 31.

Table 31 – 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	Format ID (FDh)								FC ID (28h)								CMND IU Length															
1	Reserved																				Category				Flags							
2	(MSB) Connection Identifier																															
3																									(LSB)							
4	Command Sequence Number																															
5	Data Length																															
6	NVM Submission Queue Entry (64 bytes)																															
21																																
22	DPS								LBADS								MS															
23	Reserved																															

Format ID: The Format ID field shall be set to FDh (see annex F).

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

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

Category: The Category field is specified in table 32.

Table 32 – Category field description

Value	Description
0000b	Not specified.
0001b	This NVMe_CMND IU SQE is associated with the Admin Queue.
1xxxb	This NVMe_CMND IU SQE is associated with an I/O Queue, where xxxb is set to the value of the I/O Command Set Selected (i.e, CSS) field in the Controller Configuration (i.e., CC) register (see NVM Express).

Flags: The Flags field bits are specified in table 33.

Table 33 – Flags field description

Bit	Description
0	Write
1	Read
2	PI Control Word Present
3 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.

If the PI Control Word Present bit is set to zero, then the DPS field, the LBADS field, and the MS field in the NVMe_CMND IU are not valid and shall be set to zero. If the PI Control Word Present bit is set to one, then the DPS field, the LBADS field, and the MS field in the NVMe_CMND IU contain values that may be used to support NVM Express end-to-end data protection. The PI Control Word Present bit shall not be set to one unless the PI control capability has been successfully negotiated using PRLI (see 4.20).

NOTE 3 – The target NVMe_Port should verify the values in the DPS field, the LBADS field, and the MS field in the NVMe_CMND IU with the corresponding namespace values. If one of these values does not match, then the target NVMe_Port should send an appropriate error in an NVMe_ERSP IU.

Connection Identifier: The Connection Identifier field identifies the specific SQ the SQE is to be submitted to, and indirectly identifies the NVMe controller in the NVM subsystem. See 4.4.

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).

Data Protection Type Settings (DPS): This field indicates the type settings for the end-to-end data protection feature on the namespace and contains the DPS field as specified in the Identify Namespace Data Structure (see NVM Express).

LBA Data Size (LBADS): This field indicates the LBA data size supported on the namespace and contains the LBA Data Size field as specified in the in the LBA Format Data Structure (see NVM Express).

Metadata Size (MS): This field indicates the number of metadata bytes provided per LBA supported on the namespace and contains the Metadata Size field as specified in the LBA Format Data Structure (see NVM Express).

9.3 NVMe_XFER_RDY IU format

The format of the NVMe_XFER_RDY IU is specified in table 34.

Table 34 – 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 has occurred (see 11.2).

9.4 NVMe_DATA IU format

9.4.1 NVMe_DATA IU format overview

The format of the NVMe_DATA IU is a standard FC Sequence with the TYPE field set to 08h and the R_CTL field set to 01h for a Data-Out action (see table 29) or Data-In action (see table 30). The payload of the Sequence corresponds to a range of sequential data in the Data Series for the NVMeoFC I/O operation. The start of the range is indicated by the Parameter field in the first frame of the Sequence. The length of the range is the Sequence payload length. Each frame in the Sequence is a continually increasing portion of the Data Series range. The Parameter field of each frame specifies the Data Series offset that corresponds to the start of the frame's payload.

The specific usage of NVMe_DATA IUs for NVMeoFC Data Series transfer is provided in 9.4.2 and 9.4.3.

9.4.2 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.

An NVMeoFC data transfer error is detected (see 11.2) if an NVMeoFC data transfer does not meet 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 received 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 detect an NVMeoFC data transfer error (see 11.2).

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.

9.4.3 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.

9.4.4 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;
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; and
e) be set to zero upon transmitting an NVMe_SR IU requesting retransmission of either an NVMe_XFER_RDY IU or an NVMe_DATA IU.

If an initiator NVMe_Port receives an NVMe_ERSP IU Transferred Data Length field value that does not 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 has occurred (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;
d) remain zero if first burst was received and discarded; and
e) be set to zero upon receiving an NVMe_SR IU requesting retransmission of either an NVMe_XFER_RDY IU or an NVMe_DATA IU.

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.

9.4.5 NVMe_DATA IU use of fill bytes (see FC-FS-6)

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 35.

Table 35 – 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_ERSP IU response rules are specified in 4.9. The format of the NVMe_ERSP IU is specified in table 36.

Table 36 – 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 37.

Table 37 – ERSP Result field values

Value	Name	Description
00h	SUCCESS	No status.
01h	INVALID FIELD	NVMe_CMND IU field is invalid.
02h	-	Reserved
03h	ILLEGAL CONNECT PARAMETERS	Connect command parameters are invalid for the NVMeoFC connection.
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 NVMe_DATA 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.

9.8 NVMe_SR IU format

The NVMe_SR IU contains a request for Sequence retransmission from the initiator NVMe_Port to the target NVMe_Port to recover from an Exchange error. The NVMe_SR IU shall only be used if the SLER capability has been successfully negotiated between the initiator NVMe_Port and the target NVMe_Port. Sequence Initiative is passed to the target NVMe_Port.

If the NVMe_SR IU requires retransmission of an NVMe_DATA IU, then the entire NVMeoFC Data Series shall be retransmitted.

A SLER qualifier shall be provided in the Parameter field of the Frame_Header for this IU.

Any Sequences transmitted due to an NVMe_SR IU request shall follow normal FC-FS-6 rules for SEQ_ID and SEQ_CNT.

The format of the NVMe_SR IU is specified in table 38.

Table 38 – NVMe_SR 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	FC ID (28h)								Opcode (01h)								Reserved								Retry R_CTL							
1	Reserved																															

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

Opcode: The Opcode field shall be set to 01h to indicate an NVMe_SR IU.

Retry R_CTL: The Retry R_CTL field specifies the NVMeoFC IU being requested for retransmission. The Retry R_CTL field encoding is as described in FC-FS-6 (i.e., Data Descriptor for an NVMe_XFER_RDY IU, Command Status for an NVMe_RSP, Extended Command Status for an NVMe_ERSP, and Solicited Data for an NVMe_DATA IU).

9.9 NVMe_SR_RSP IU format

The NVMe_SR_RSP IU contains a response to an NVMe_SR IU to recover from an Exchange error. The NVMe_SR_RSP IU shall only be used if the SLER capability has been successfully negotiated between the initiator NVMe_Port and the target NVMe_Port. The target NVMe_Port shall transmit the NVMe_SR_RSP IU before retransmitting sequences requested by the NVMe_SR IU. The NVMe_SR_RSP IU shall not pass Sequence Initiative to the initiator NVMe_Port unless the Status field is set to a non-zero value.

If the target NVMe_Port determines that the S_ID, OX_ID, RX_ID, or SLER qualifier fields are inconsistent for the Exchange, then it shall respond with the Status field set to “Invalid qualifiers”.

The format of the NVMe_SR_RSP IU is specified in table 39.

Table 39 – NVMe_SR_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	FC ID (28h)								Opcode (01h)								Reserved								Status							
1	Reserved																															

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

Opcode: The Opcode field shall be set to 01h to indicate an NVMe_SR_RSP IU.

Status: The Status field indicates if the target NVMe_Port is able to perform the retransmission requested by the initiator NVMe_Port. A value of zero indicates that the target NVMe_Port shall perform the retransmission following transmission of this NVMe_SR_RSP IU. If a non-zero value is returned, then the initiator NVMe_Port shall terminate the Exchange using ABTS-LS. The Status field values are specified in table 40.

Table 40 – Status field values

Value	Name	Description
00h	Accepted	The retransmission request shall be performed.
01h	Invalid FC ID	The FC ID field does not contain a valid value.
02h	Invalid Opcode	The Opcode field does not contain a valid value.
03h	Logical error	The NVMe_SR IU request and payload content is invalid or logically inconsistent for the conditions present.
04h	Invalid qualifiers	The Exchange qualifiers are inconsistent.
09h	Unable to perform retransmission request	The recipient of an NVMe_SR IU request is unable to perform the request.
Others	-	Reserved

Editor's Note: Prefer Status field Name in all CAPS, and 09h = NOT ACCEPTED.

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 4 – 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 41.

Table 41 – 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	Transport Service ID (TRSVCID) (32 bytes)																															
15																																
16 to 63	Reserved																															
64	NVMe Qualified Name (SUBNQN) (256 bytes)																															
127																																
128	Transport Address (TRADDR) (256 bytes)																															
191																																
192	Transport Specific Address Subtype (TSAS) (256 bytes)																															
255																																

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.

10.1.3 Keep Alive settings

Keep Alive functionality (see NVM Express) is not required for NVMeoFC. NVMeoFC does not impose any limitations on the minimum and maximum Keep Alive Timeout value.

If Keep Alive is used for NVMeoFC, then the Keep Alive Timeout value minimum should be set large enough to account for any transient fabric interconnect failures between the host and controller.

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-6. See 11.2 for a list of these mechanisms.

11.2 Error detection

An initiator NVMe_Port shall detect the following errors:

a) a Sequence error (see FC-FS-6);
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;
f) an NVMeoFC data transfer error (see 4.9 and clause 9); and
g) an NVMeoFC transport error (see 4.9).

A target NVMe_Port shall detect the following errors:

a) a Sequence error (see FC-FS-6);
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; and
c) an NVMeoFC data transfer error (see clause 9).

If the SLER capability has been successfully negotiated and an error is detected that can be recovered using Sequence retransmission, then the methods specified in 11.7 may be used to complete the Exchange. Otherwise, upon detection of an error:

a) if the Exchange is open, the detecting NVMe_Port shall transmit an ABTS-LS to terminate the Exchange and recover the associated Exchange resources (see 11.3);
b) if the Exchange is not open, the NVMeoFC connection and NVMeoFC association that were associated with the Exchange shall be terminated as specified in clause 4.

11.3 Exchange level termination and resource recovery using ABTS-LS

11.3.1 ABTS-LS overview

ABTS-LS is an FC-FS-6 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 the NVMe I/O Exchange. Refer to the actions specified in clause 4 for termination of FC-NVMe transport connections and associations.

Initiator NVMe_Ports 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-6 allows ABTS-LS to be transmitted by an Nx_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 or BA_RJT 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.

The NVMe_Port terminating the Exchange may reclaim the Exchange resources without requiring the reception of a BA_ACC or BA_RJT or performing second level error recovery (see 11.4) if the NVMe_Port is an initiator NVMe_Port and one of the Exchange recovery options are met as described in 4.3.3, or if the NVMe_Port is a target NVMe_Port and one of the Exchange recovery options are met as described in 4.3.5.

If the NVMe_Port is an initiator NVMe_Port and a BA_ACC or BA_RJT response is not received from the corresponding NVMe_Port within two times R_A_TOV, and the Exchange resource has not been recovered by other options as described in 4.3.3, then second level error recovery (see 11.4) shall be performed.

If the NVMe_Port is a target NVMe_Port and a BA_ACC or BA_RJT response is not received from the corresponding NVMe_Port within two times R_A_TOV, and the Exchange resource has not been recovered by other options as described in 4.3.5, 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 an 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_Port 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 may 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 an 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, and the Exchange resource has not been recovered by other options as described in 4.3.3 for an initiator NVMe_Port and in 4.3.5 for a target NVMe_Port, then the NVMe_Port may transmit the ABTS-LS again, attempt other retry operations allowed by FC-FS-6, 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.

11.6 Clearing effects of NVMeoFC, FC-FS-6, and FC-LS-4 actions

11.6.1 Overview of clearing effects

Certain Fibre Channel link actions and NVMeoFC operations have clearing effects on more than just a single I/O operation and may influence connection, association, and login states.

11.6.2 Clearing effects of N_Port Logout (LOGO)

Once a login is established between an initiator NVMe_Port and target NVMe_Port, transmission or reception of a LOGO ELS has these clearing effects between the two NVMe_Ports;

a) open NVMeoFC Exchanges are terminated;
b) active NVMeoFC connections are terminated;
c) active NVMeoFC associations are terminated;
d) Process Login (PRLI) ELS parameters are cleared; and
e) N_Port Login (PLOGI) ELS service parameters set to default values (see FC-LS-4).

11.6.3 Clearing effects of Process Logout (PRLO, TPRLO)

Once a process login is established between an initiator NVMe_Port and target NVMe_Port, transmission or reception of a PRLO ELS with Type Code 0x28 has these clearing effects between the two NVMe_Ports;

a) open NVMeoFC Exchanges are terminated using ABTS-LS (see 11.3);
b) active NVMeoFC connections are terminated;
c) active NVMeoFC associations are terminated; and
d) Process Login (PRLI) ELS parameters are cleared.

If a target NVMe_Port receives a Third Party Process Logout (TPRLO) ELS, the actions listed above shall be performed for all initiator NVMe_Ports that have an established Process Login with the recipient target NVMe_Port when the Global Process Logout bit is set to one. If the Global Process Logout bit is set to zero, then the actions shall be performed for the designated initiator NVMe_Port. A target NVMe_Port should transmit a PRLO ELS to all additional logged-in initiator NVMe_Ports that are logged out as a result of processing a TPRLO ELS with the Global Process Logout bit set to one. The PRLO ELS(es) may be transmitted before or after transmitting the LS_ACC for the TPRLO ELS.

11.6.4 Clearing effects of N_Port Login (PLOGI)

Once a login is established between an initiator NVMe_Port and target NVMe_Port, transmission or reception of another PLOGI causes an implicit logout followed by relogin with the new service parameters, if the PLOGI is successful.

Implicit logout has the same clearing effects as N_Port Logout (see 11.6.2).

The new login parameters take effect at the responder when an LS_ACC is transmitted and at the originator when the LS_ACC is received.

11.6.5 Clearing effects of Process Login (PRLI)

Once a Process Login is established between an initiator NVMe_Port and target NVMe_Port, transmission or reception of a PRLI ELS causes an implicit Process Logout, followed by Process Login with the new parameters. The implicit Process Logout has the these clearing effects;

a) open NVMeoFC Exchanges are terminated using ABTS-LS (see 11.3);
b) active NVMeoFC connections are terminated;
c) active NVMeoFC associations are terminated; and
d) Process Login (PRLI) ELS parameters are cleared.

The new Process Login parameters take effect at the responder when an LS_ACC is transmitted and at the originator when the LS_ACC is received.

11.6.6 Clearing effects of Disconnect NVMe_LS

A Disconnect NVMe_LS causes an NVMeoFC association to be terminated. Transmission or reception of a Disconnect NVMe_LS also cause all related Exchanges and connections to be terminated (see 4.3.2 and 4.3.4).

11.6.7 Clearing effects of NVMe Controller Reset or Shutdown

If the NVMe host performs a Controller Reset or Shutdown action (see NVM Express), the following clearing actions take place for the association and the connections related to the controller being reset;

a) open NVMeoFC Exchanges are terminated using ABTS-LS (see 11.3);
b) active NVMeoFC connections are terminated; and
c) the corresponding NVMeoFC association is terminated.

Controller Reset and Shutdown do not cause explicit or implicit Process Logout or N_Port Logout.

11.7 NVMeoFC Sequence level error recovery

11.7.1 Overview

NVMeoFC Sequence level error recovery (SLER) allows Exchange recovery from transmission errors by using Sequence retransmission. The SLER capability shall be successfully negotiated between the initiator NVMe_Port and the target NVMe_Port before using:

a) the FLUSH, FLUSH_RSP, or RED Basic Link Service commands;
b) the NVMe_SR IU;
c) the NVMe_SR_RSP IU; or
d) any recovery operations specified in this clause.

Target NVMe_Ports that successfully negotiated SLER shall set the RX_ID field value to a unique value other than FFFFh for each Exchange.

If the SLER capability has not been successfully negotiated, and a FLUSH command, FLUSH_RSP, RED command, NVMe_SR IU, or NVMe_SR_RSP IU is received, it shall be discarded.

If the SLER capability has been successfully negotiated, the target NVMe_Port shall close the Exchange only if:

a) an NVMe_CONF IU is received for the Exchange;
b) a FLUSH_RSP with the Open Exchange bit set to zero is transmitted (see 11.7.5.2);
c) a FLUSH_RSP with the Open Exchange bit set to zero is received (see 11.7.5.10);
d) IR_TOV expires after the most recent NVMe_RSP IU or NVMe_ERSP IU was transmitted to the initiator NVMe_Port;
e) a new NVMe_CMND IU is received with the same S_ID and OX_ID field values; or
f) the Exchange is aborted.

The target NVMe_Port shall retain the Exchange information until the Exchange is closed.

The Exchange information retained shall include:

a) data transfer information;
b) data descriptors; and

c) NVMe_RSP IU or NVMe_ERSP IU information.

11.7.2 FLUSH command usage

See FC-FS-6 for a description of the FLUSH Basic Link Service request Sequence.

The FLUSH command is issued by an NVMe_Port to determine what the FLUSH Recipient NVMe_Port's view of the Exchange state is. The FLUSH Recipient responds with a FLUSH_RSP (see 11.7.3). The FLUSH Initiator uses this information to determine if Sequence Level Error Recovery is needed.

NVMeoFC specific usage of the FLUSH command is as follows:

a) the FLUSH command may be transmitted by an initiator NVMe_Port;
b) the FLUSH command may be transmitted by a target NVMe_Port, but shall only be used to determine if an NVMe_CONF has been transmitted;
c) the SLER qualifier shall be provided in the Parameter field of the Frame_Header; and
d) if the Halt Transmission bit is set to one, then the target NVMe_Port ceases processing the Exchange as described in FC-FS-6, and an NVMe_SR IU is the FC-4 specific event necessary for the target NVMe_Port to continue processing the Exchange.

11.7.3 FLUSH_RSP usage

See FC-FS-6 for a description of the FLUSH_RSP Basic Link Service reply Sequence.

FLUSH_RSP contains a response to a FLUSH command indicating the FLUSH Recipient NVMe_Port's view of the Exchange state.

NVMeoFC specific usage of a FLUSH_RSP transmitted by a target NVMe_Port is as follows:

a) the target NVMe_Port shall use the Exchange S_ID, OX_ID, RX_ID and the SLER qualifier in the FLUSH command to determine whether an Exchange is open;
b) the Parameter field of the Frame_Header shall be set to zero;
c) if the Halt Transmission bit is set to one in the FLUSH command, then the target NVMe_Port ceases processing the Exchange as described in FC-FS-6, and an NVMe_SR IU is the FC-4 specific event necessary for the target NVMe_Port to continue processing the Exchange;
d) if the target NVMe_Port sets the Sequence Error bit to one or sets the Waiting for FC-4 Specific Event bit to one in the FLUSH_RSP, the FC-4 specific event necessary for the target NVMe_Port to continue processing the Exchange is an NVMe_SR IU;
e) the FC-4 Capability bit in the FLUSH_RSP payload shall be set to zero;
f) the FC4INFO field in the FLUSH_RSP payload shall be set to the R_CTL value in the most recent NVMe IU transmitted by the target NVMe_Port for the Exchange. If no NVMe IUs have been transmitted by the target NVMe_Port for the Exchange, then the FC4INFO field shall be set to 00h; and
g) the FC4VALUE field in the FLUSH_RSP payload indicates the number of data bytes transmitted or received for the Exchange and shall be set to the byte count of transferred data (see 9.4.4).

NVMeoFC specific usage of a FLUSH_RSP transmitted by an initiator NVMe_Port is as follows:

a) the initiator NVMe_Port shall use the Exchange S_ID, OX_ID, RX_ID, and the SLER qualifier in the FLUSH command to determine whether an Exchange is open;
b) the Parameter field of the Frame_Header shall be set to zero;

c) the initiator NVMe_Port shall ignore the Halt Transmission bit value in the FLUSH command and process the Halt Transmission bit as having the value zero;
d) the FC-4 Capability bit in the FLUSH_RSP payload shall be set to zero; and
e) the FC4INFO field and the FC4VALUE field in the FLUSH_RSP payload shall be set to zero.

11.7.4 RED command usage

See FC-FS-6 for a description of the RED Basic Link Service request Sequence.

The RED command may be transmitted by a target NVMe_Port to indicate to an initiator NVMe_Port that a Sequence error was detected in an open Exchange.

For NVMeoFC, the FC-4 specific event associated with a RED command is an NVMe_SR IU.

11.7.5 Using information from FLUSH and FLUSH_RSP to perform Sequence retransmission

11.7.5.1 Polling Exchange state with FLUSH commands

A FLUSH command is periodically transmitted by the initiator NVMe_Port to poll each outstanding Exchange to determine if a command is progressing properly and if any Sequences have been received incorrectly. Timing of polling with the FLUSH command is controlled by FLUSH_TOV (see table 42). FLUSH_TOV should be long enough that processing the transfers of Sequence Initiative in the Exchange and completing the Exchange occur before FLUSH_TOV expires. If FLUSH_TOV expires, then a FLUSH command is transmitted by the initiator NVMe_Port. The information returned by the target NVMe_Port in the FLUSH_RSP payload is compared with the expected state information known by the initiator NVMe_Port. If the information is inconsistent (e.g., the target NVMe_Port indicates it sent an NVMe_RSP but the initiator NVMe_Port did not receive it), indicating that a link error occurred, then error recovery actions may be performed to complete the Exchange. Error detection and recovery procedures are specified in 11.7.5.

If the target NVMe_Port responds to the NVMe_SR IU with a non-zero Status field value in the NVMe_SR_RSP IU, the initiator NVMe_Port shall terminate the Exchange using ABTS-LS.

A target NVMe_Port may send a FLUSH command to an initiator NVMe_Port, but only to determine if an NVMe_CONF IU has been transmitted (see 11.7.5.10).

11.7.5.2 NVMe_CMND IU recovery

A received NVMe_CMND IU is detected by reception of a FLUSH_RSP with the Open Exchange bit set to one. If a FLUSH_RSP is received with the Open Exchange bit set to one and the Sequence Initiative State bit is set to one (i.e., the target NVMe_Port holds Sequence Initiative), then the command is in process and no recovery is needed at this time.

A lost NVMe_CMND IU is detected by reception of a FLUSH_RSP with the Open Exchange bit set to zero. If a FLUSH_RSP is transmitted with the Open Exchange bit set to zero, then the target NVMe_Port shall close the Exchange. If a FLUSH_RSP is received with the Open Exchange bit set to zero, then the initiator NVMe_Port shall close the Exchange and then retransmit the NVMe_CMND IU with the SEQ_CNT field set to zero.

11.7.5.3 NVMe_XFER_RDY IU recovery

For a write operation, if the initiator NVMe_Port receives an NVMe_XFER_RDY IU for an Exchange after transmitting a FLUSH command and before the FLUSH_RSP is received, then normal processing of the NVMe I/O operation for that Exchange shall continue and the contents of the FLUSH_RSP shall be ignored.

For a write operation, a lost NVMe_XFER_RDY IU is detected by reception of a FLUSH_RSP with the Sequence Initiative State bit is set to zero and the FC4INFO field is set to 05h (i.e., an NVMe_XFER_RDY IU). If the FLUSH_RSP indicates that an NVMe_XFER_RDY IU was sent by the target NVMe_Port, but was not received by the initiator NVMe_Port, then the initiator NVMe_Port shall transmit an NVMe_SR IU to request retransmission of an NVMe_XFER_RDY IU.

For a write operation, if the target NVMe_Port receives a request for retransmission of an NVMe_XFER_RDY IU, the target NVMe_Port shall:

1) transmit an NVMe_SR_RSP IU for the NVMe_SR IU; and
2) if the Status field in the NVMe_SR_RSP IU is set to 00h, then the target NVMe_Port shall transmit an NVMe_XFER_RDY IU with the Relative Offset field set to zero.

Upon receipt of the NVMe_XFER_RDY IU, the data transfer for the NVMe I/O operation restarts, even if the NVMe_SR_RSP IU is not received. All NVMe_DATA IUs for the Exchange shall be retransmitted.

11.7.5.4 NVMe_RSP or NVMe_ERSP IU recovery

A lost NVMe_RSP IU or NVMe_ERSP IU is detected if:

a) the FLUSH_RSP indicates that an NVMe_RSP IU or NVMe_ERSP IU was sent by the target NVMe_Port (i.e., the FC4INFO value is 07h or 08h), but the initiator NVMe_Port has not received the NVMe_RSP IU or NVMe_ERSP IU; and
b) the Exchange is performing a read operation and the FLUSH_RSP FC4VALUE field in the FLUSH_RSP is equal to the initiator NVMe_Port's transferred byte count (i.e., all data sent by the target NVMe_Port was successfully received by the initiator NVMe_Port).

If all data was not successfully received for a read operation, the initiator NVMe_Port shall perform NVMe_DATA IU recovery for read operations as specified in 11.7.5.6.

If a lost NVMe_RSP IU or NVMe_ERSP IU is detected, then the initiator NVMe_Port shall transmit an NVMe_SR IU to request retransmission of the NVMe_RSP IU or NVMe_ERSP IU as indicated in the FC4INFO field of the FLUSH_RSP.

If the target NVMe_Port receives a request for retransmission of an NVMe_RSP IU or NVMe_ERSP IU, then the target NVMe_Port shall:

1) transmit an NVMe_SR_RSP IU for the NVMe_SR IU; and
2) retransmit the NVMe_RSP IU or NVMe_ERSP IU in a new Sequence.

Upon receipt of the NVMe_RSP IU or NVMe_ERSP IU, the NVMe I/O operation continues or completes normally, even if the NVMe_SR_RSP IU is lost.

11.7.5.5 NVMe_DATA IU recovery for write operations

For a write operation, if the FC4VALUE field in a FLUSH_RSP is less than the initiator NVMe_Port's transferred byte count or the Sequence Error bit is set to one (i.e., an NVMe_DATA IU was sent by the initiator NVMe_Port, but some data was not received by the target NVMe_Port), then the initiator NVMe_Port shall transmit an NVMe_SR IU to request retransmission of an NVMe_XFER_RDY IU to retransmit the Data Series.

For a write operation, if the target NVMe_Port receives a request for retransmission of an NVMe_XFER_RDY IU, the target NVMe_Port shall:

1) discard the Sequence in error and any NVMe_DATA IUs already received for the Exchange;
2) transmit an NVMe_SR_RSP IU for the NVMe_SR IU; and
3) if the Status field in the NVMe_SR_RSP IU is set to 00h, then the target NVMe_Port shall transmit an NVMe_XFER_RDY IU with the Relative Offset field set to zero.

Upon receipt of the NVMe_XFER_RDY IU, the data transfer for the NVMe I/O operation restarts, even if the NVMe_SR_RSP IU is not received. The NVMe_DATA IU shall be retransmitted in a new Sequence. If continuously increasing sequence count is being used, then the SEQ_CNT field value in the retransmitted NVMe_DATA IU shall be zero or continue following the continuously increasing sequence count rules (see FC-FS-6). All NVMe_DATA IUs for the Exchange shall be retransmitted.

11.7.5.6 NVMe_DATA IU recovery for read operations

One or more lost NVMe_DATA IU frames for read operations (i.e., data was sent by the target NVMe_Port but not successfully received by the initiator NVMe_Port) are detected if:

a) the FLUSH_RSP has the Sequence Initiative State bit set to zero and the FC4VALUE field is greater than the initiator NVMe_Port's transferred byte count; or
b) a Sequence error on the read data was detected by the initiator NVMe_Port.

If one or more lost NVMe_DATA IU frames for read operations are detected, then:

1) if the initiator NVMe_Port does not hold Sequence Initiative, then the initiator NVMe_Port shall send a FLUSH command with the Halt Transmission bit set to one and wait for the FLUSH_RSP before further processing of the Exchange. While the initiator NVMe_Port is waiting for the FLUSH_RSP, the initiator NVMe_Port shall continue FLUSH command polling (see 11.7.5.1) with the Halt Transmission bit set to one and shall respond to ABTS-LS commands; and
2) the initiator NVMe_Port shall transmit an NVMe_SR IU to request retransmission of all NVMe_DATA IUs.

For a read operation, if the target NVMe_Port receives a request for retransmission of an NVMe_DATA IU, then the target NVMe_Port shall:

1) transmit the NVMe_SR_RSP IU for the NVMe_SR IU;
2) transmit all NVMe_DATA IUs in new Sequences; and
3) complete the Exchange in a normal manner, including retransmitting the NVMe_RSP IU or NVMe_ERSP IU.

If continuously increasing sequence count is being used, then the SEQ_CNT field value in the retransmitted NVMe_DATA IU shall be zero or continue following the continuously increasing sequence count rules (see FC-FS-6). After an NVMe_DATA IU is successfully received, the NVMe I/O operation continues normally, even if the NVMe_SR_RSP IU is lost.

11.7.5.7 NVMe_SR IU recovery

If an NVMe_SR IU has been sent to the target NVMe_Port and an NVMe_SR_RSP IU or the retransmitted Sequence have not been received by the initiator NVMe_Port, FLUSH command polling is used to determine if the NVMe_SR IU has been lost.

A lost NVMe_SR IU is detected if:

a) the FLUSH_RSP WSE bit is set to one (i.e., the target NVMe_Port is waiting for an NVMe_SR IU and the NVMe_SR IU has been lost); or
b) the FC4INFO field in the FLUSH_RSP is not set to 0Ah (i.e., an NVMe_SR_RSP IU) and the Sequence Initiative State bit is set to zero (i.e., either the NVMe_SR IU has been lost or the NVMe_SR_RSP IU and the retransmitted Sequence have been lost).

To recover a lost NVMe_SR IU, the NVMe_SR IU is retransmitted in a new Sequence from the initiator NVMe_Port to the target NVMe_Port.

11.7.5.8 NVMe_SR_RSP IU recovery

If an NVMe_SR IU is transmitted and the retransmitted Sequence is received by the initiator NVMe_Port without receiving an NVMe_SR_RSP IU, then no recovery of the lost NVMe_SR_RSP IU is performed. The Exchange proceeds using the retransmitted Sequence.

If an NVMe_SR IU is transmitted and a subsequent FLUSH command receives a FLUSH_RSP with the Sequence Initiative State bit set to zero and the FC4INFO field set to 0Ah (i.e., an NVMe_SR_RSP IU), then this indicates that an NVMe_SR_RSP IU with the Status field set to a non-zero value was lost. The initiator NVMe_Port shall issue an ABTS-LS to terminate the Exchange and no recovery of the lost NVMe_SR_RSP IU is performed.

11.7.5.9 RED recovery

The Responder Error Detected (RED) command may be transmitted by a target NVMe_Port to indicate that a Sequence error was detected for a write operation on an open Exchange. This may result in quicker error recovery for the write operation than waiting for a FLUSH command.

Upon reception of a RED command, the initiator NVMe_Port shall transmit an NVMe_SR IU to request retransmission of an NVMe_XFER_RDY IU to retransmit the Data Series.

For a write operation, if the target NVMe_Port receives a request for retransmission of an NVMe_XFER_RDY IU, the target NVMe_Port shall:

1) discard the Sequence in error and any NVMe_DATA IUs already received for the Exchange;
2) transmit an NVMe_SR_RSP IU for the NVMe_SR IU; and
3) if the Status field in the NVMe_SR_RSP IU is set to 00h, then the target NVMe_Port shall transmit an NVMe_XFER_RDY IU with the Relative Offset field set to zero.

Upon receipt of the NVMe_XFER_RDY IU, the data transfer for the NVMe I/O operation restarts, even if the NVMe_SR_RSP IU is not received. The NVMe_DATA IU shall be retransmitted in a new Sequence. If continuously increasing sequence count is being used, then the SEQ_CNT field value in the retransmitted NVMe_DATA IU shall be zero or continue following the continuously increasing sequence count rules (see FC-FS-6). All NVMe_DATA IUs for the Exchange shall be retransmitted.

11.7.5.10 NVMe_CONF IU recovery

If the target NVMe_Port has requested that the initiator NVMe_Port transmit an NVMe_CONF IU (see 4.10), then the target NVMe_Port should poll the initiator NVMe_Port by transmitting a FLUSH command to the initiator NVMe_Port to determine if the NVMe_CONF has been transmitted. Timing of polling with the FLUSH command is controlled by FLUSH_TOV (see table 42). The target NVMe_Port shall wait FLUSH_TOV before transmitting the first FLUSH command.

A lost NVMe_CONF IU is detected if the FLUSH_RSP from the initiator NVMe_Port has the Open Exchange bit set to zero (i.e., the Exchange has been closed) and the NVMe_CONF IU has not been received by the target NVMe_Port. If a lost NVMe_CONF IU is detected, then the target NVMe_Port shall assume that the NVMe_CONF IU was sent and close the Exchange.

If the initiator NVMe_Port has not sent the NVMe_CONF IU for this Exchange before a FLUSH command is received, then the FLUSH_RSP shall have the Open Exchange bit set to one, indicating the Exchange is still open. In this case the target NVMe_Port shall wait FLUSH_TOV and, if the Exchange has not been closed, transmit another FLUSH command. The target NVMe_Port shall repeat this process until the Exchange is closed or until an NVMe_SR IU is received for the Exchange.

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.

Timers used for operation and recovery are summarized in table 42.

Table 42 – Timers summary

Timer	Implementation		Description	Default Value	Ref
	Initiator NVMe_Port	Target NVMe_Port			
R_A_TOV	M	M	Resource_Allocation_ Timeout Value	see FC-FS-6	12.2
IR_TOV	n/a	M	Initiator Response Timeout Value	If SLER Supported bit is set to zero: 2 s[a] If SLER Supported bit is set to one: ≥ FLUSH_TOV + (2 × R_A_TOV) + 1 s	12.3
FLUSH_TOV	M	O	Flush Timeout Value	≥ 2s (minimum)	12.4
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 used as the timeout value for determining when to reinstate a Recovery_Qualifier (see FC-FS-6).

An NVMe_Port may immediately reinstate a Recovery_Qualifier after receiving a BA_ACC or BA_RJT 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.

12.4 Flush Timeout Value (FLUSH_TOV)

FLUSH_TOV is used by an NVMe_Port to provide a minimum polling interval for the FLUSH command. The use of FLUSH_TOV by a target NVMe_Port is specified in NVMe_CONF IU recovery (see 11.7.5.10).

At least one FLUSH_TOV period shall elapse between transmission of an NVMe_CMND and the first polling for Exchange status with the FLUSH command by an initiator NVMe_Port.

For each Exchange, the initiator NVMe_Port starts or restarts the FLUSH_TOV timer upon:

a) the initiator NVMe_Port transferring Sequence Initiative to the target NVMe_Port without closing or aborting the Exchange; or
b) transmission of the FLUSH command.

The initiator NVMe_Port stops the FLUSH_TOV timer upon:

a) receiving Sequence Initiative from the target NVMe_Port; or
b) closing 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

The NVMe_CMND IU payload is divided into an NVMeoFC header area (i.e., word 0 to word 5), an NVM Express SQE area (i.e., word 6 to word 21), and an NVMeoFC trailer area (i.e., word 22 to word 23). Fields in the NVMeoFC header and NVMeoFC trailer areas are in big-endian format. The fields in the NVM Express SQE area are in little-endian format.

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	Format ID (FDh)								FC ID (28h)								(MSB) CMND IU Length (LSB)															
1	Reserved																				Category				Flags							
2	(MSB)								Connection Identifier																							
3																									(LSB)							
4	(MSB)								Command Sequence Number																(LSB)							
5	(MSB)								Data Length																(LSB)							
6	Opcode (OPC)								PSDT		Reserved				FUSE		(LSB) Command Identifier (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				SGL Descriptor Sub Type			
16	Command Dword 10																															
17	Command Dword 11																															
18	Command Dword 12																															
19	Command Dword 13																															
20	Command Dword 14																															
21	Command Dword 15																															
22	DPS								LBADS								(MSB) MS (LSB)															
23	Reserved																															

A.3 NVMe_ERSP IU payload

Table A.2 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 an NVMe_CMND IU. The SQE within the IU contains the command. The IU indicates the 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 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 an NVMe_CMND IU. The SQE within the IU contains the command. The IU indicates the 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 connection of the command;
2) The initiator NVMe_Port transmits an NVMe_DATA IU containing an initial burst of data for the write operation. The data for the 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 be 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 NVMe_Port. 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-4);
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, 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 I/O Connection 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.

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.

C.1.3 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

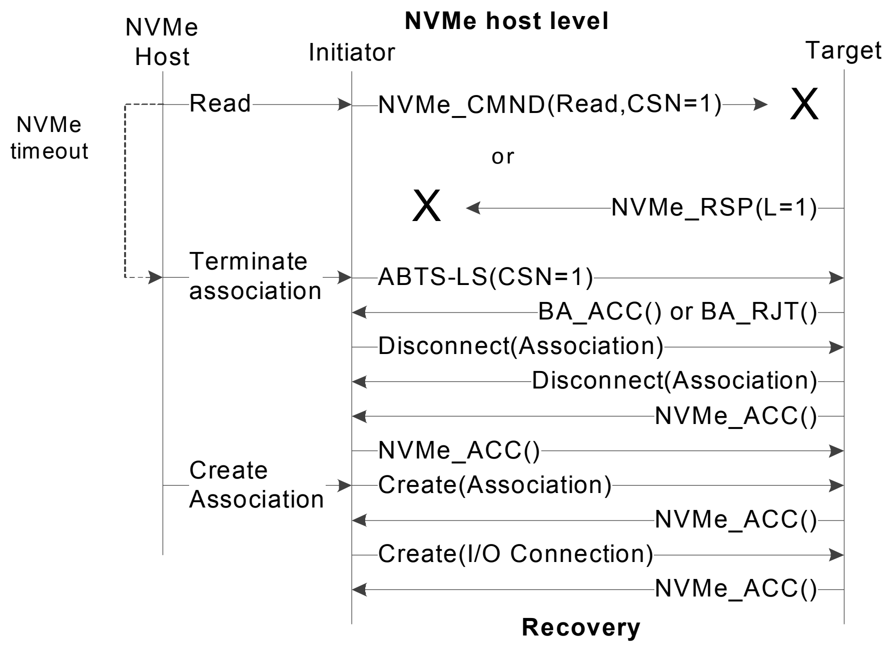
D.1 Overview

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 association termination is shown in figure D.1.



After NVMe timeout occurs the NVMe host terminates the association and creates a new association. The Disconnect request also aborts all outstanding Exchanges using ABTS-LS.

Figure D.1 - NVMe_CMND lost or NVMe_RSP lost with association termination

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

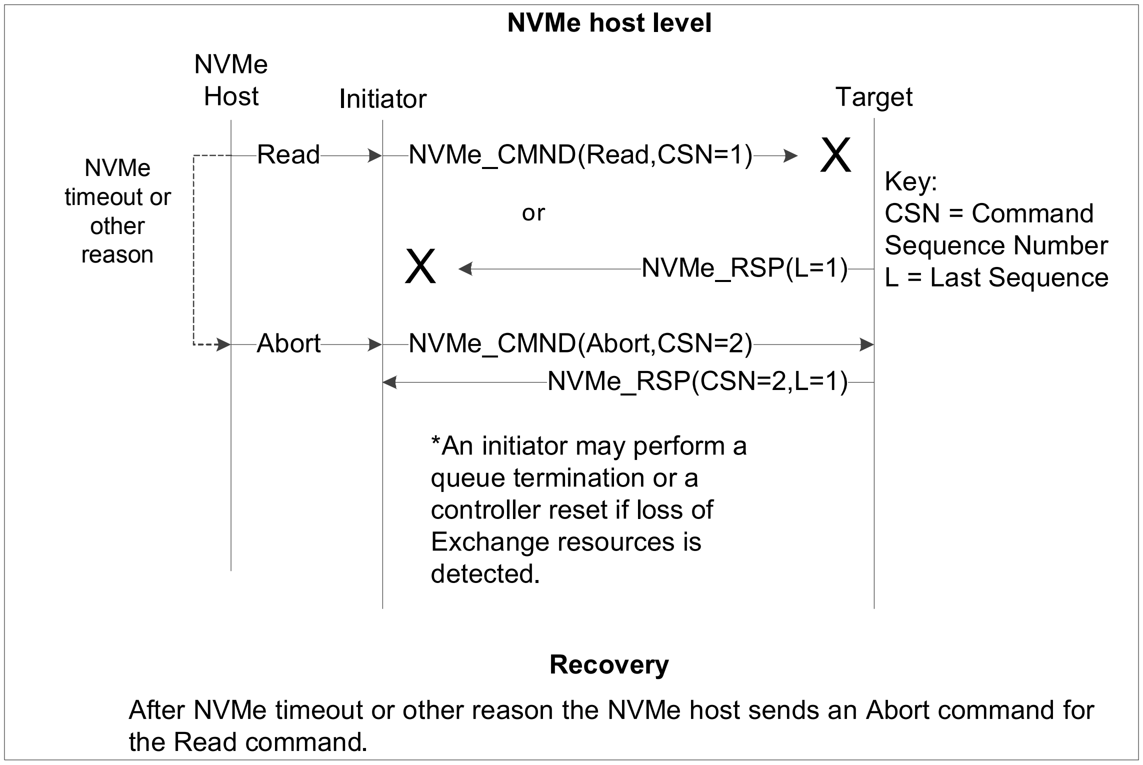

NVMe host level
NVMe Host
Initiator
Target
NVMe timeout or other reason
Read
NVMe_CMND(Read,CSN=1)
X
or
X
NVMe_RSP(L=1)
Key:
CSN = Command Sequence Number
L = Last Sequence
Abort
NVMe_CMND(Abort,CSN=2)
NVMe_RSP(CSN=2,L=1)
*An initiator may perform a queue termination or a controller reset if loss of Exchange resources is detected.
Recovery
After NVMe timeout or other reason the NVMe host sends an Abort command for the Read command.

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.

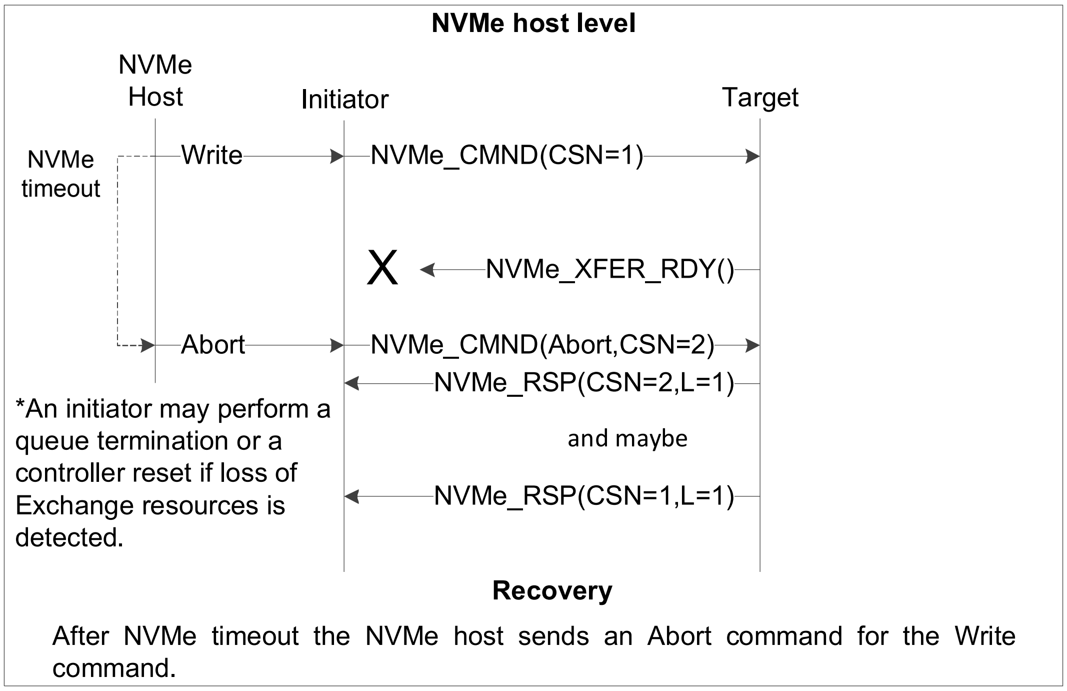

NVMe host level
NVMe Host
Initiator
Target
NVMe timeout
Write
NVMe_CMND(CSN=1)
X
NVMe_XFER_RDY()
Abort
NVMe_CMND(Abort,CSN=2)
NVMe_RSP(CSN=2,L=1)
and maybe
NVMe_RSP(CSN=1,L=1)
*An initiator may perform a queue termination or a controller reset if loss of Exchange resources is detected.
Recovery
After NVMe timeout the NVMe host sends an Abort command for the Write command.

Figure D.3 - NVMe_XFER_RDY lost with NVM Abort

Annex E
(informative)
Sequence level error detection and recovery examples

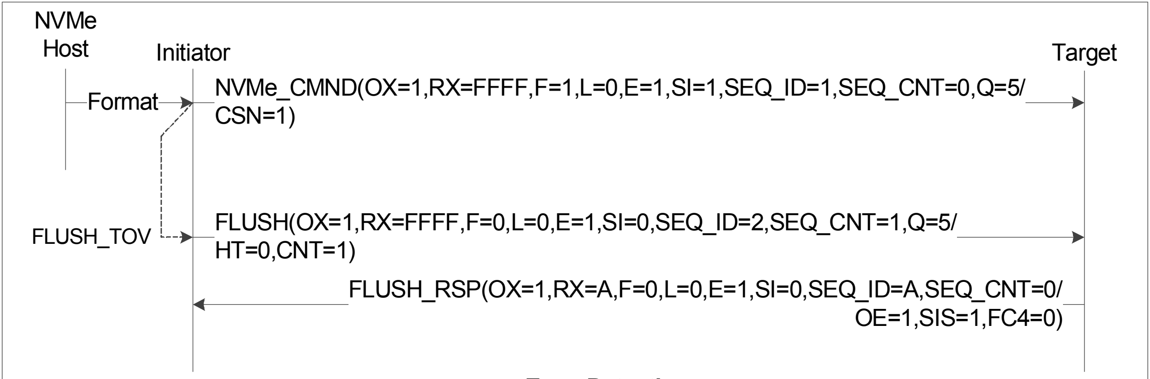
E.1 Overview

This informative annex diagrams various SLER procedures for NVMe_Ports conforming to this standard. The conventions for the diagrams are shown in table E.1.

Table E.1 - Diagram conventions

Convention	Meaning
⟶	Class 3 frame.
X	Frame lost or dropped.
/	Payload separator
Initiator	initiator NVMe_Port
Target	target NVMe_Port
CSN	Command Sequence Number
F	First_Sequence bit
L	Last_Sequence bit
E	End_Sequence bit
SI	Sequence Initiative bit
RO	Relative Offset field
Q	SLER qualifier field
HT	FLUSH Halt Transmission bit
CNT	FLUSH Count field
SIS	FLUSH_RSP Sequence Initiative State bit
OE	FLUSH_RSP Open Exchange bit
WSE	FLUSH_RSP Waiting for FC-4 Specific Event bit
FC4	FLUSH_RSP FC4INFO field

An example of a long running NVMe_CMND is shown in figure E.1.


NVMe Host
Initiator
Target
Format
NVMe_CMND(OX=1,RX=FFFF,F=1,L=0,E=1,SI=1,SEQ_ID=1,SEQ_CNT=0,Q=5/ CSN=1)
FLUSH_TOV
FLUSH(OX=1,RX=FFFF,F=0,L=0,E=1,SI=0,SEQ_ID=2,SEQ_CNT=1,Q=5/ HT=0,CNT=1)
FLUSH_RSP(OX=1,RX=A,F=0,L=0,E=1,SI=0,SEQ_ID=A,SEQ_CNT=0/ OE=1,SIS=1,FC4=0)

Error Detection

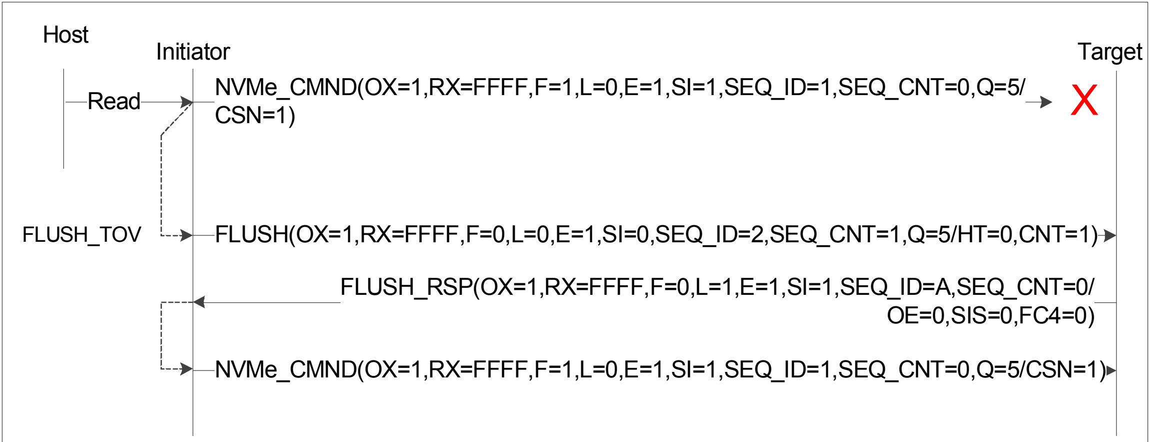
FLUSH timeout occurs after transmitting the NVMe_CMND and a FLUSH BLS is sent to determine the status of the Exchange.

Error Recovery

The FLUSH_RSP indicates the Exchange is open and the target NVMe_Port holds Sequence Initiative. No error recovery is required.

Figure E.1 - Long running NVMe_CMND

An example of a lost NVMe_CMND IU is shown in figure E.2.


Host
Initiator
Target
Read
NVMe_CMND(OX=1,RX=FFFF,F=1,L=0,E=1,SI=1,SEQ_ID=1,SEQ_CNT=0,Q=5/ CSN=1)
X
FLUSH_TOV
FLUSH(OX=1,RX=FFFF,F=0,L=0,E=1,SI=0,SEQ_ID=2,SEQ_CNT=1,Q=5/HT=0,CNT=1)
FLUSH_RSP(OX=1,RX=FFFF,F=0,L=1,E=1,SI=1,SEQ_ID=A,SEQ_CNT=0/ OE=0,SIS=0,FC4=0)
NVMe_CMND(OX=1,RX=FFFF,F=1,L=0,E=1,SI=1,SEQ_ID=1,SEQ_CNT=0,Q=5/CSN=1)

Error Detection

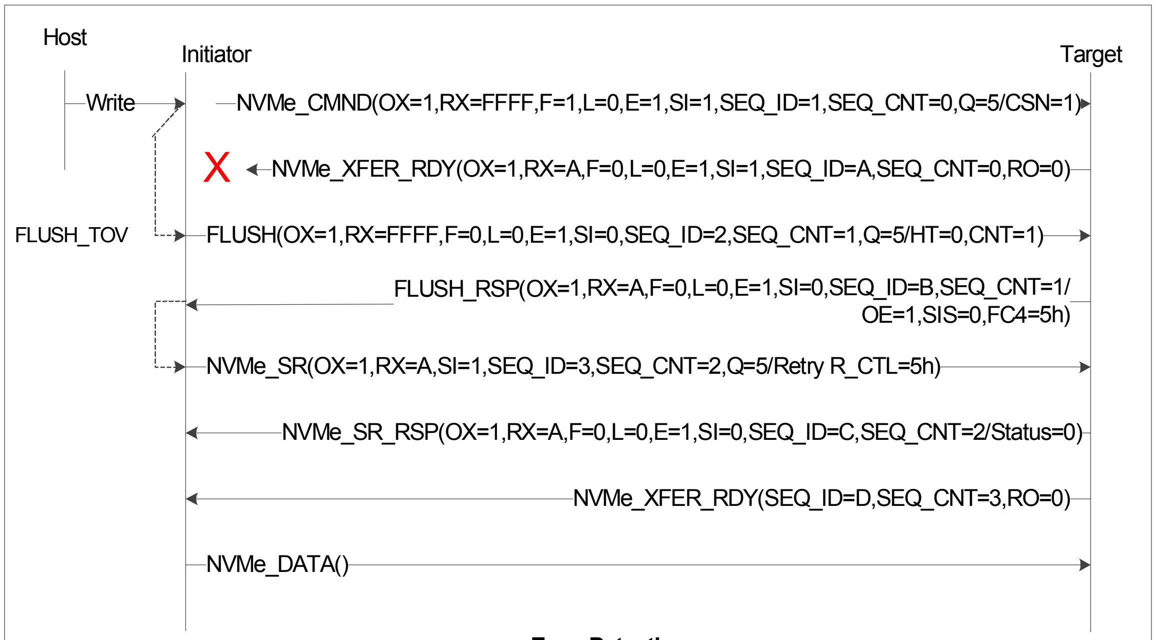
FLUSH timeout occurs after transmitting the NVMe_CMND and a FLUSH BLS is sent to determine the status of the Exchange.

Error Recovery

The FLUSH_RSP indicates the Exchange is not open at the target, and the Exchange is closed. The NVMe_CMND IU is retransmitted using the same OX_ID, SLER qualifier, and CSN.

Figure E.2 - Lost NVMe_CMND IU

An example of a lost NVMe_XFER_RDY IU is shown in figure E.3.


Host
Initiator
Target
Write
NVMe_CMND(OX=1,RX=FFFF,F=1,L=0,E=1,SI=1,SEQ_ID=1,SEQ_CNT=0,Q=5/CSN=1)
X
NVMe_XFER_RDY(OX=1,RX=A,F=0,L=0,E=1,SI=1,SEQ_ID=A,SEQ_CNT=0,RO=0)
FLUSH_TOV
FLUSH(OX=1,RX=FFFF,F=0,L=0,E=1,SI=0,SEQ_ID=2,SEQ_CNT=1,Q=5/HT=0,CNT=1)
FLUSH_RSP(OX=1,RX=A,F=0,L=0,E=1,SI=0,SEQ_ID=B,SEQ_CNT=1/ OE=1,SIS=0,FC4=5h)
NVMe_SR(OX=1,RX=A,SI=1,SEQ_ID=3,SEQ_CNT=2,Q=5/Retry R_CTL=5h)
NVMe_SR_RSP(OX=1,RX=A,F=0,L=0,E=1,SI=0,SEQ_ID=C,SEQ_CNT=2/Status=0)
NVMe_XFER_RDY(SEQ_ID=D,SEQ_CNT=3,RO=0)
NVMe_DATA()

Error Detection

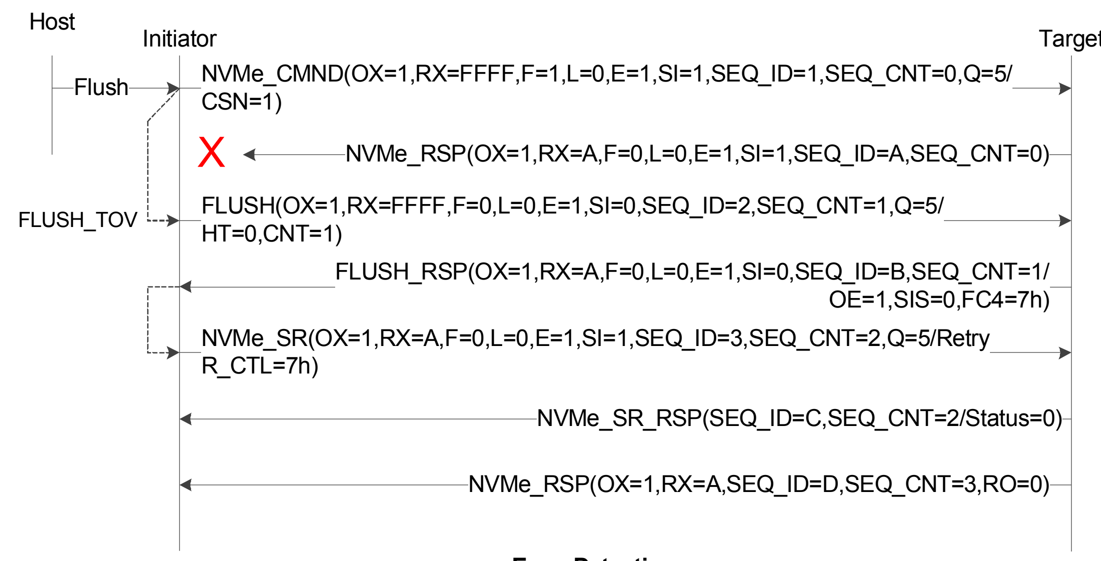
FLUSH timeout occurs after transmitting the NVMe_CMND and a FLUSH BLS is sent to determine the status of the Exchange.

Error Recovery

The FLUSH_RSP indicates the initiator NVMe_Port holds Sequence Initiative and the Exchange is open. The initiator NVMe_Port transmits an NVMe_SR IU specifying the NVMe_XFER_RDY be resent. The target NVMe_Port retransmits the NVMe_XFER_RDY using Relative Offset zero. A new SEQ_ID is used for retransmission of the NVMe_XFER_RDY.

Figure E.3 - Lost NVMe_XFER_RDY IU

An example of a lost NVMe_RSP IU is shown in figure E.4.


Host
Initiator
Target
Flush
FLUSH_TOV
NVMe_CMND(OX=1,RX=FFFF,F=1,L=0,E=1,SI=1,SEQ_ID=1,SEQ_CNT=0,Q=5/CSN=1)
X
NVMe_RSP(OX=1,RX=A,F=0,L=0,E=1,SI=1,SEQ_ID=A,SEQ_CNT=0)
FLUSH(OX=1,RX=FFFF,F=0,L=0,E=1,SI=0,SEQ_ID=2,SEQ_CNT=1,Q=5/HT=0,CNT=1)
FLUSH_RSP(OX=1,RX=A,F=0,L=0,E=1,SI=0,SEQ_ID=B,SEQ_CNT=1/OE=1,SIS=0,FC4=7h)
NVMe_SR(OX=1,RX=A,F=0,L=0,E=1,SI=1,SEQ_ID=3,SEQ_CNT=2,Q=5/Retry R_CTL=7h)
NVMe_SR_RSP(SEQ_ID=C,SEQ_CNT=2/Status=0)
NVMe_RSP(OX=1,RX=A,SEQ_ID=D,SEQ_CNT=3,RO=0)

Error Detection

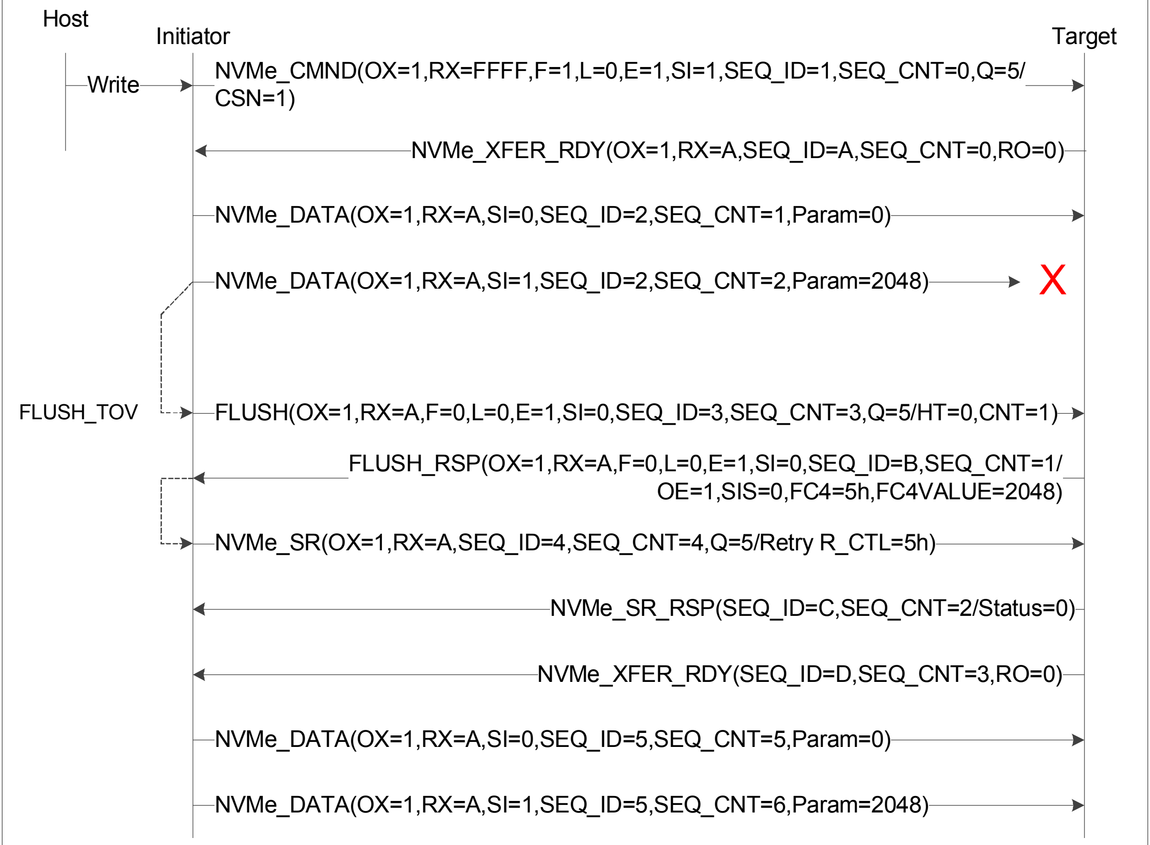
FLUSH timeout occurs after transmitting the NVMe_CMND and a FLUSH BLS is sent to determine the status of the Exchange.

Error Recovery

The FLUSH_RSP indicates the initiator NVMe_Port holds Sequence Initiative and the Exchange is open. The initiator NVMe_Port transmits an NVMe_SR IU specifying the NVMe_RSP be resent. The target NVMe_Port retransmits the NVMe_RSP using a new SEQ_ID.

Figure E.4 - Lost NVMe_RSP IU

An example of lost write data, last frame of Sequence is shown in figure E.5.


Host
Initiator
Target
Write
NVMe_CMND(OX=1,RX=FFFF,F=1,L=0,E=1,SI=1,SEQ_ID=1,SEQ_CNT=0,Q=5/CSN=1)
NVMe_XFER_RDY(OX=1,RX=A,SEQ_ID=A,SEQ_CNT=0,RO=0)
NVMe_DATA(OX=1,RX=A,SI=0,SEQ_ID=2,SEQ_CNT=1,Param=0)
NVMe_DATA(OX=1,RX=A,SI=1,SEQ_ID=2,SEQ_CNT=2,Param=2048)
X
FLUSH_TOV
FLUSH(OX=1,RX=A,F=0,L=0,E=1,SI=0,SEQ_ID=3,SEQ_CNT=3,Q=5/HT=0,CNT=1)
FLUSH_RSP(OX=1,RX=A,F=0,L=0,E=1,SI=0,SEQ_ID=B,SEQ_CNT=1/OE=1,SIS=0,FC4=5h,FC4VALUE=2048)
NVMe_SR(OX=1,RX=A,SEQ_ID=4,SEQ_CNT=4,Q=5/Retry R_CTL=5h)
NVMe_SR_RSP(SEQ_ID=C,SEQ_CNT=2/Status=0)
NVMe_XFER_RDY(SEQ_ID=D,SEQ_CNT=3,RO=0)
NVMe_DATA(OX=1,RX=A,SI=0,SEQ_ID=5,SEQ_CNT=5,Param=0)
NVMe_DATA(OX=1,RX=A,SI=1,SEQ_ID=5,SEQ_CNT=6,Param=2048)

Error Detection

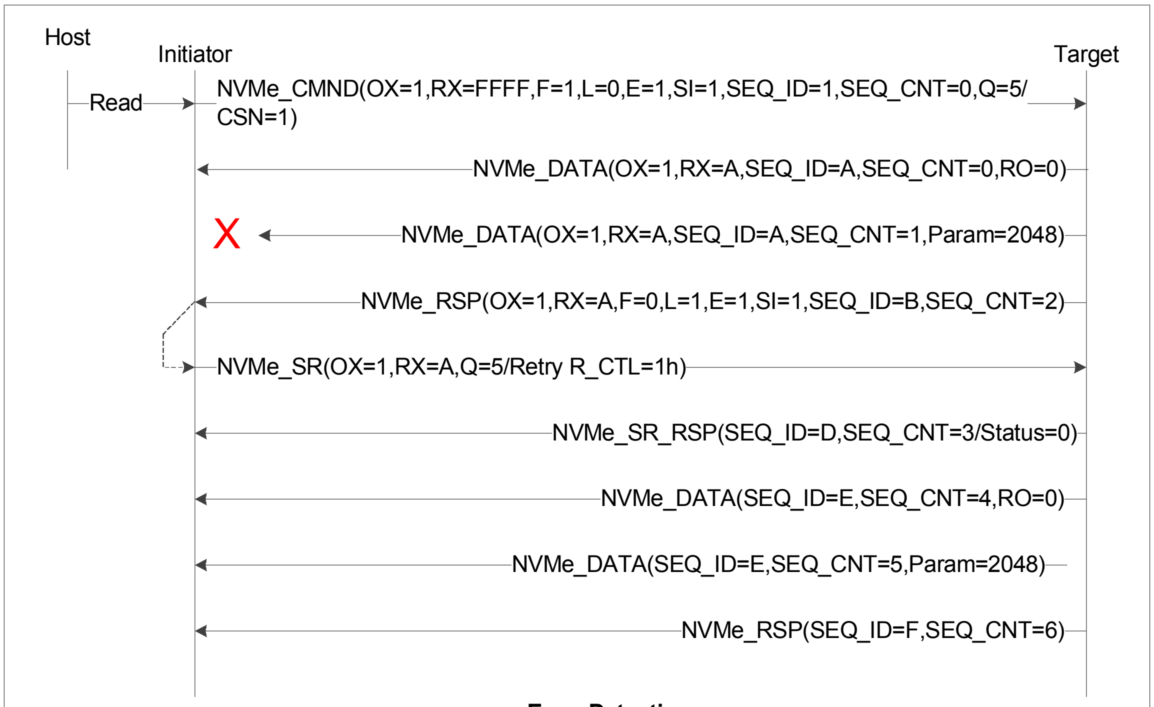
FLUSH timeout occurs after transmitting the NVMe_DATA and a FLUSH BLS is sent to determine the status of the Exchange.

Error Recovery

The FLUSH_RSP indicates the initiator NVMe_Port holds Sequence Initiative, the Exchange is open, and 2048 bytes have been received by the target NVMe_Port. The initiator NVMe_Port transmits an NVMe_SR IU specifying the NVMe_XFER_RDY be resent. The target NVMe_Port retransmits the NVMe_XFER_RDY with Relative Offset set to zero. The initiator NVMe_Port retransmits the data.

Figure E.5 - Lost write data, last frame of Sequence

An example of lost read data, last frame of Sequence is shown in figure E.6.


Host
Initiator
Target
Read
NVMe_CMND(OX=1,RX=FFFF,F=1,L=0,E=1,SI=1,SEQ_ID=1,SEQ_CNT=0,Q=5/CSN=1)
NVMe_DATA(OX=1,RX=A,SEQ_ID=A,SEQ_CNT=0,RO=0)
X
NVMe_DATA(OX=1,RX=A,SEQ_ID=A,SEQ_CNT=1,Param=2048)
NVMe_RSP(OX=1,RX=A,F=0,L=1,E=1,SI=1,SEQ_ID=B,SEQ_CNT=2)
NVMe_SR(OX=1,RX=A,Q=5/Retry R_CTL=1h)
NVMe_SR_RSP(SEQ_ID=D,SEQ_CNT=3/Status=0)
NVMe_DATA(SEQ_ID=E,SEQ_CNT=4,RO=0)
NVMe_DATA(SEQ_ID=E,SEQ_CNT=5,Param=2048)
NVMe_RSP(SEQ_ID=F,SEQ_CNT=6)

Error Detection

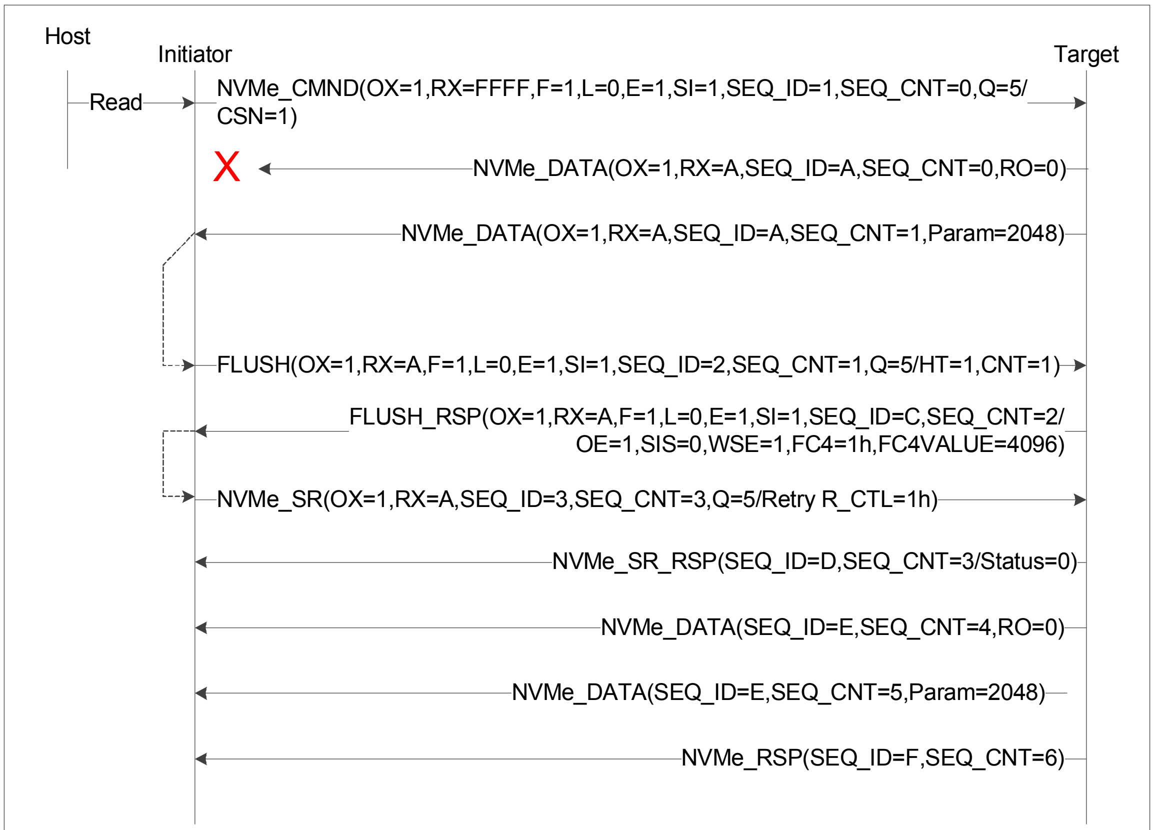
The NVMe_RSP is received and the initiator NVMe_Port detects a Sequence error.

Error Recovery

The initiator NVMe_Port has Sequence Initiative and transmits an NVMe_SR IU specifying the data be resent. The target NVMe_Port retransmits the data from Relative Offset zero.

Figure E.6 - Lost read data, last frame of Sequence

An example of lost read data, not last frame of Sequence is shown in figure E.7.



Host
Initiator
Target
Read
NVMe_CMND(OX=1,RX=FFFF,F=1,L=0,E=1,SI=1,SEQ_ID=1,SEQ_CNT=0,Q=5/CSN=1)
X
NVMe_DATA(OX=1,RX=A,SEQ_ID=A,SEQ_CNT=0,RO=0)
NVMe_DATA(OX=1,RX=A,SEQ_ID=A,SEQ_CNT=1,Param=2048)
FLUSH(OX=1,RX=A,F=1,L=0,E=1,SI=1,SEQ_ID=2,SEQ_CNT=1,Q=5/HT=1,CNT=1)
FLUSH_RSP(OX=1,RX=A,F=1,L=0,E=1,SI=1,SEQ_ID=C,SEQ_CNT=2/OE=1,SIS=0,WSE=1,FC4=1h,FC4VALUE=4096)
NVMe_SR(OX=1,RX=A,SEQ_ID=3,SEQ_CNT=3,Q=5/Retry R_CTL=1h)
NVMe_SR_RSP(SEQ_ID=D,SEQ_CNT=3/Status=0)
NVMe_DATA(SEQ_ID=E,SEQ_CNT=4,RO=0)
NVMe_DATA(SEQ_ID=E,SEQ_CNT=5,Param=2048)
NVMe_RSP(SEQ_ID=F,SEQ_CNT=6)

Error Detection

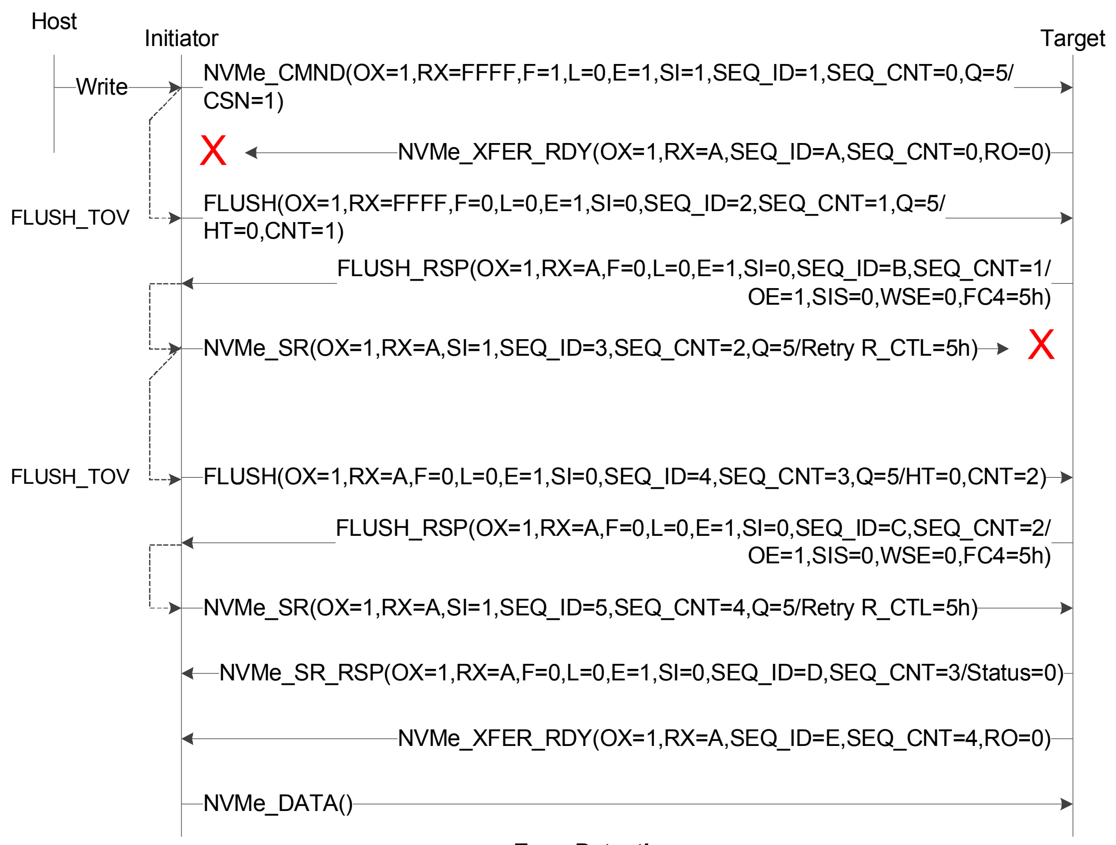
The initiator NVMe_Port detects a Sequence error and a FLUSH BLS with the HT bit set to one is sent to transfer Sequence Initiative back to the initiator NVMe_Port for error recovery.

Error Recovery

The FLUSH_RSP transfers Sequence Initiative to the initiator NVMe_Port. The initiator NVMe_Port transmits an NVMe_SR IU specifying the data be resent. The target NVMe_Port retransmits the data from Relative Offset zero.

Figure E.7 - Lost read data, not last frame of Sequence

An example of a lost NVMe_SR IU is shown in figure E.8.


Host
Initiator
Target
Write
NVMe_CMND(OX=1,RX=FFFF,F=1,L=0,E=1,SI=1,SEQ_ID=1,SEQ_CNT=0,Q=5/CSN=1)
X
NVMe_XFER_RDY(OX=1,RX=A,SEQ_ID=A,SEQ_CNT=0,RO=0)
FLUSH_TOV
FLUSH(OX=1,RX=FFFF,F=0,L=0,E=1,SI=0,SEQ_ID=2,SEQ_CNT=1,Q=5/HT=0,CNT=1)
FLUSH_RSP(OX=1,RX=A,F=0,L=0,E=1,SI=0,SEQ_ID=B,SEQ_CNT=1/OE=1,SIS=0,WSE=0,FC4=5h)
NVMe_SR(OX=1,RX=A,SI=1,SEQ_ID=3,SEQ_CNT=2,Q=5/Retry R_CTL=5h)
X
FLUSH_TOV
FLUSH(OX=1,RX=A,F=0,L=0,E=1,SI=0,SEQ_ID=4,SEQ_CNT=3,Q=5/HT=0,CNT=2)
FLUSH_RSP(OX=1,RX=A,F=0,L=0,E=1,SI=0,SEQ_ID=C,SEQ_CNT=2/OE=1,SIS=0,WSE=0,FC4=5h)
NVMe_SR(OX=1,RX=A,SI=1,SEQ_ID=5,SEQ_CNT=4,Q=5/Retry R_CTL=5h)
NVMe_SR_RSP(OX=1,RX=A,F=0,L=0,E=1,SI=0,SEQ_ID=D,SEQ_CNT=3/Status=0)
NVMe_XFER_RDY(OX=1,RX=A,SEQ_ID=E,SEQ_CNT=4,RO=0)
NVMe_DATA()

Error Detection

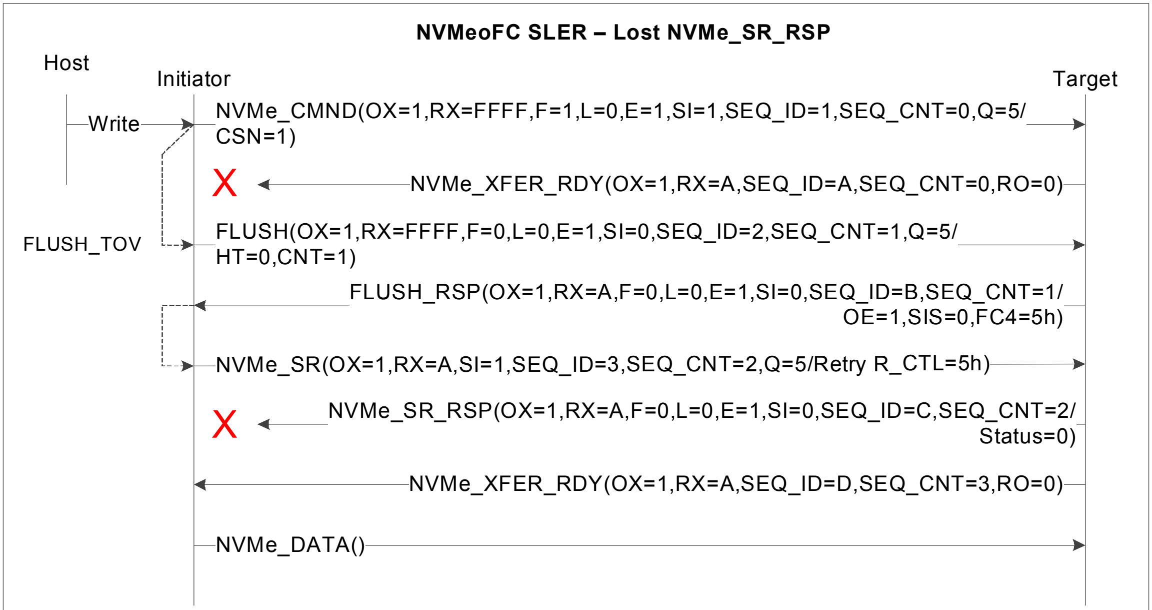
FLUSH timeout occurs after transmitting the NVMe_SR and a FLUSH BLS is sent to determine the status of the Exchange.

Error Recovery

The FLUSH_RSP indicates the initiator NVMe_Port holds Sequence Initiative, the Exchange is open, and a NVMe_XFER_RDY was transmitted. The initiator NVMe_Port transmits an NVMe_SR IU specifying the NVMe_XFER_RDY be resent. The target NVMe_Port retransmits the NVMe_XFER_RDY with Relative Offset set to zero. A new SEQ_ID is used for retransmission of the NVMe_XFER_RDY.

Figure E.8 - Lost NVMe_SR IU

An example of a lost NVMe_SR_RSP IU is shown in figure E.9.


NVMeoFC SLER – Lost NVMe_SR_RSP
Host
Initiator
Target
Write
NVMe_CMND(OX=1,RX=FFFF,F=1,L=0,E=1,SI=1,SEQ_ID=1,SEQ_CNT=0,Q=5/CSN=1)
X
NVMe_XFER_RDY(OX=1,RX=A,SEQ_ID=A,SEQ_CNT=0,RO=0)
FLUSH_TOV
FLUSH(OX=1,RX=FFFF,F=0,L=0,E=1,SI=0,SEQ_ID=2,SEQ_CNT=1,Q=5/HT=0,CNT=1)
FLUSH_RSP(OX=1,RX=A,F=0,L=0,E=1,SI=0,SEQ_ID=B,SEQ_CNT=1/OE=1,SIS=0,FC4=5h)
NVMe_SR(OX=1,RX=A,SI=1,SEQ_ID=3,SEQ_CNT=2,Q=5/Retry R_CTL=5h)
X
NVMe_SR_RSP(OX=1,RX=A,F=0,L=0,E=1,SI=0,SEQ_ID=C,SEQ_CNT=2/Status=0)
NVMe_XFER_RDY(OX=1,RX=A,SEQ_ID=D,SEQ_CNT=3,RO=0)
NVMe_DATA()

Error Detection

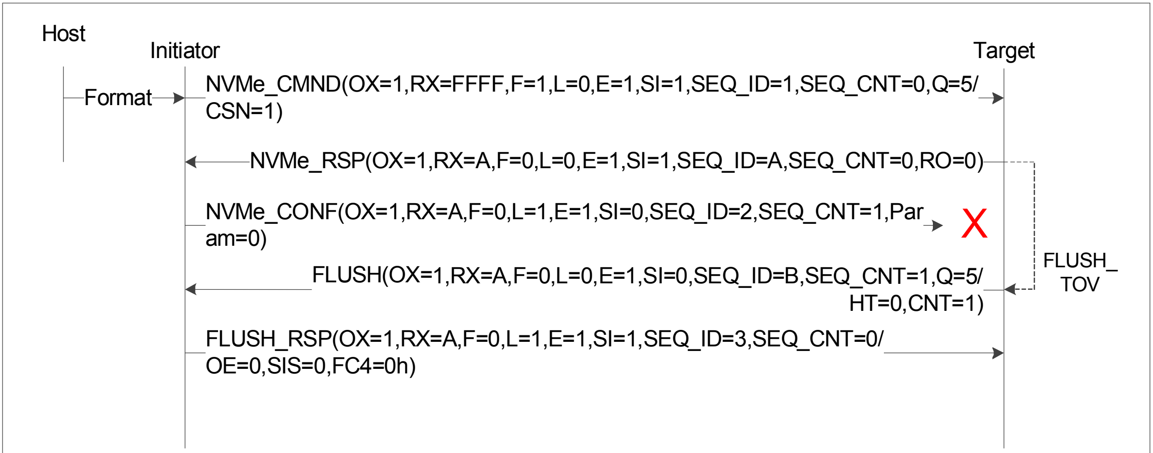
FLUSH timeout occurs after transmitting the NVMe_CMND and a FLUSH BLS is sent to determine the status of the Exchange.

Error Recovery

The FLUSH_RSP indicates the initiator NVMe_Port holds Sequence Initiative, the Exchange is open, and an NVMe_XFER_RDY was transmitted. The initiator NVMe_Port transmits an NVMe_SR IU specifying the NVMe_XFER_RDY be resent. The NVMe_SR_RSP is lost. The target NVMe_Port retransmits the NVMe_XFER_RDY with Relative Offset set to zero. A new SEQ_ID is used for retransmission of the NVMe_XFER_RDY. Since the NVMe_XFER_RDY is received, no recovery of the NVMe_SR_RSP is needed.

Figure E.9 - Lost NVMe_SR_RSP IU

An example of a lost NVMe_CONF IU is shown in figure E.10.


Host
Initiator
Target
Format
NVMe_CMND(OX=1,RX=FFFF,F=1,L=0,E=1,SI=1,SEQ_ID=1,SEQ_CNT=0,Q=5/CSN=1)
NVMe_RSP(OX=1,RX=A,F=0,L=0,E=1,SI=1,SEQ_ID=A,SEQ_CNT=0,RO=0)
NVMe_CONF(OX=1,RX=A,F=0,L=1,E=1,SI=0,SEQ_ID=2,SEQ_CNT=1,Param=0)
X
FLUSH_TOV
FLUSH(OX=1,RX=A,F=0,L=0,E=1,SI=0,SEQ_ID=B,SEQ_CNT=1,Q=5/HT=0,CNT=1)
FLUSH_RSP(OX=1,RX=A,F=0,L=1,E=1,SI=1,SEQ_ID=3,SEQ_CNT=0/OE=0,SIS=0,FC4=0h)

Error Detection

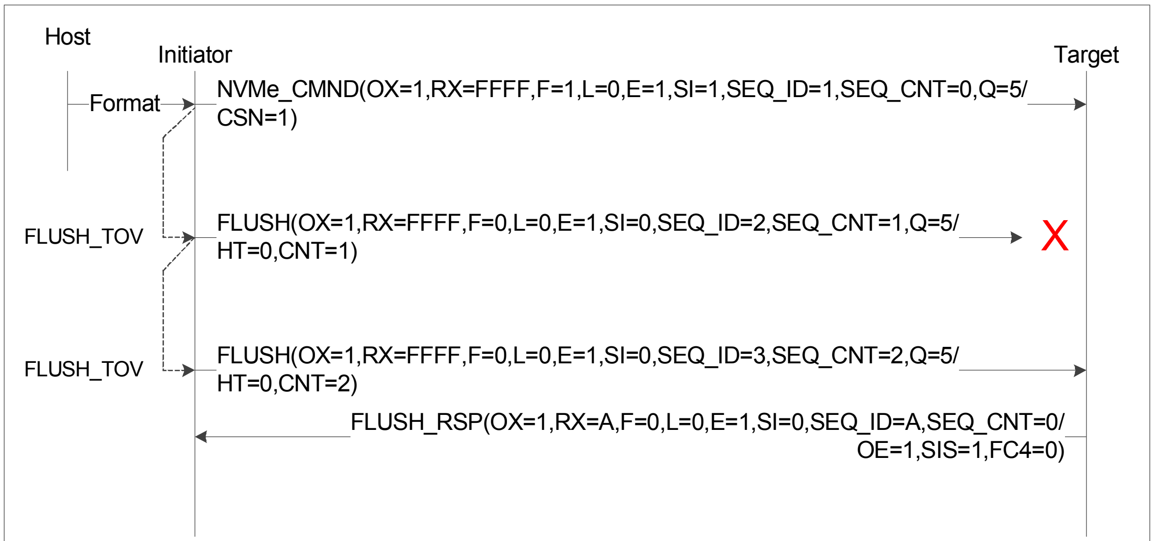
FLUSH timeout occurs after transmitting the NVMe_RSP and a FLUSH BLS is sent to determine the status of the Exchange.

Error Recovery

The FLUSH_RSP indicates the Exchange is closed, and the target NVMe_Port releases Exchange resources.

Figure E.10 - Lost NVMe_CONF IU

An example of a lost FLUSH BLS is shown in figure E.11.


Host
Initiator
Target
Format
NVMe_CMND(OX=1,RX=FFFF,F=1,L=0,E=1,SI=1,SEQ_ID=1,SEQ_CNT=0,Q=5/CSN=1)
FLUSH_TOV
FLUSH(OX=1,RX=FFFF,F=0,L=0,E=1,SI=0,SEQ_ID=2,SEQ_CNT=1,Q=5/HT=0,CNT=1)
X
FLUSH_TOV
FLUSH(OX=1,RX=FFFF,F=0,L=0,E=1,SI=0,SEQ_ID=3,SEQ_CNT=2,Q=5/HT=0,CNT=2)
FLUSH_RSP(OX=1,RX=A,F=0,L=0,E=1,SI=0,SEQ_ID=A,SEQ_CNT=0/OE=1,SIS=1,FC4=0)

Error Detection

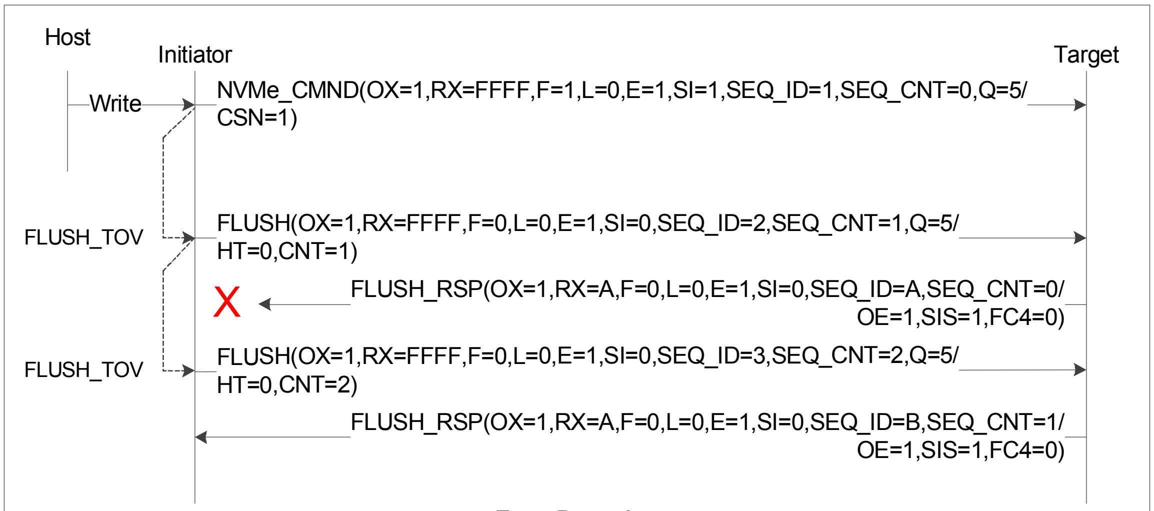
FLUSH timeout occurs after transmitting the NVMe_CMND and a FLUSH BLS is transmitted to determine the status of the Exchange. FLUSH timeout occurs after transmitting the FLUSH, and another FLUSH BLS is transmitted.

Error Recovery

The FLUSH_RSP indicates the Exchange is open and the target NVMe_Port holds Sequence Initiative. No error recovery is required.

Figure E.11 - Lost FLUSH BLS

An example of a lost FLUSH_RSP BLS is shown in figure E.12.


Host
Initiator
Target
Write
NVMe_CMND(OX=1,RX=FFFF,F=1,L=0,E=1,SI=1,SEQ_ID=1,SEQ_CNT=0,Q=5/CSN=1)
FLUSH_TOV
FLUSH(OX=1,RX=FFFF,F=0,L=0,E=1,SI=0,SEQ_ID=2,SEQ_CNT=1,Q=5/HT=0,CNT=1)
X
FLUSH_RSP(OX=1,RX=A,F=0,L=0,E=1,SI=0,SEQ_ID=A,SEQ_CNT=0/OE=1,SIS=1,FC4=0)
FLUSH_TOV
FLUSH(OX=1,RX=FFFF,F=0,L=0,E=1,SI=0,SEQ_ID=3,SEQ_CNT=2,Q=5/HT=0,CNT=2)
FLUSH_RSP(OX=1,RX=A,F=0,L=0,E=1,SI=0,SEQ_ID=B,SEQ_CNT=1/OE=1,SIS=1,FC4=0)

Error Detection

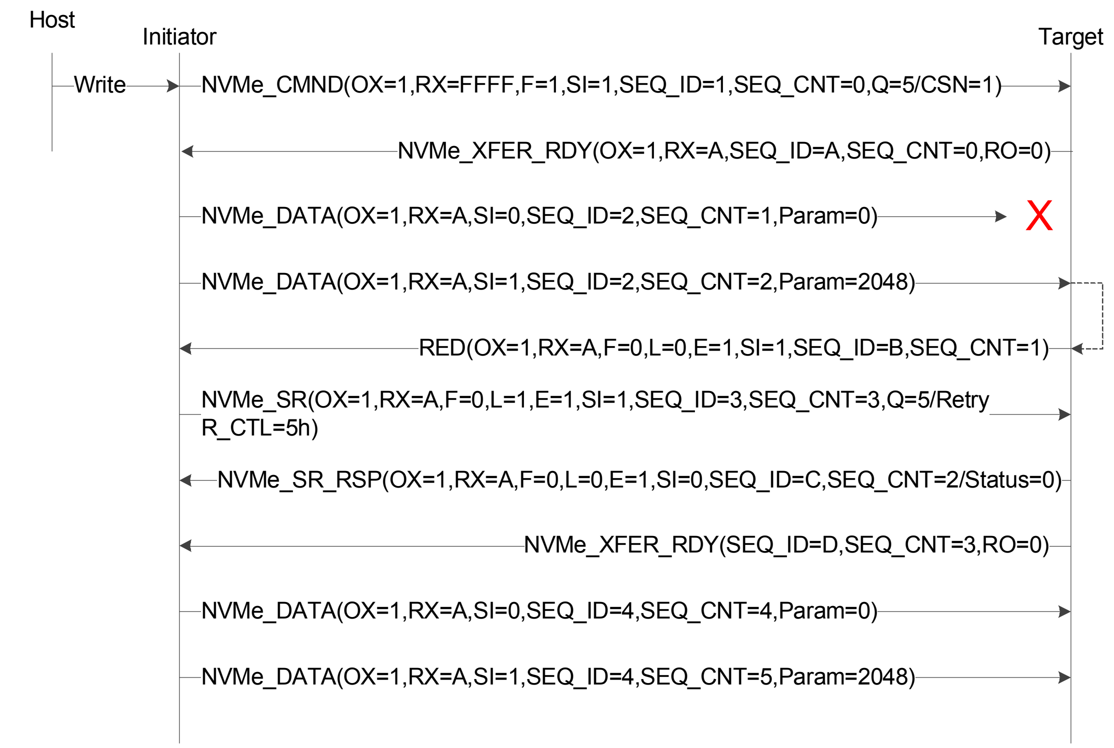
FLUSH timeout occurs after transmitting the NVMe_CMND and a FLUSH BLS is transmitted to determine the status of the Exchange. FLUSH timeout occurs after transmitting the FLUSH, and another FLUSH BLS is transmitted.

Error Recovery

The FLUSH_RSP indicates the Exchange is open and the target NVMe_Port holds Sequence Initiative. No error recovery is required.

Figure E.12 - Lost FLUSH_RSP BLS

An example of RED BLS usage is shown in figure E.13.


Host
Initiator
Target
Write
NVMe_CMND(OX=1,RX=FFFF,F=1,SI=1,SEQ_ID=1,SEQ_CNT=0,Q=5/CSN=1)
NVMe_XFER_RDY(OX=1,RX=A,SEQ_ID=A,SEQ_CNT=0,RO=0)
NVMe_DATA(OX=1,RX=A,SI=0,SEQ_ID=2,SEQ_CNT=1,Param=0)
X
NVMe_DATA(OX=1,RX=A,SI=1,SEQ_ID=2,SEQ_CNT=2,Param=2048)
RED(OX=1,RX=A,F=0,L=0,E=1,SI=1,SEQ_ID=B,SEQ_CNT=1)
NVMe_SR(OX=1,RX=A,F=0,L=1,E=1,SI=1,SEQ_ID=3,SEQ_CNT=3,Q=5/Retry R_CTL=5h)
NVMe_SR_RSP(OX=1,RX=A,F=0,L=0,E=1,SI=0,SEQ_ID=C,SEQ_CNT=2/Status=0)
NVMe_XFER_RDY(SEQ_ID=D,SEQ_CNT=3,RO=0)
NVMe_DATA(OX=1,RX=A,SI=0,SEQ_ID=4,SEQ_CNT=4,Param=0)
NVMe_DATA(OX=1,RX=A,SI=1,SEQ_ID=4,SEQ_CNT=5,Param=2048)

Error Detection

After detecting a Sequence error the target transmits a RED BLS.

Error Recovery

After receiving the RED BLS, the initiator NVMe_Port transmits an NVMe_SR IU specifying the NVMe_XFER_RDY be resent. The target NVMe_Port retransmits the NVMe_XFER_RDY with Relative Offset set to zero. The initiator NVMe_Port retransmits the data.

Figure E.13 - RED BLS usage example

Annex F
(informative)
NVMeoFC Identification

F.1 NVMeoFC code points

F.1.1 Overview

Since other protocols use the FCP data handling protocol (see FCP-4 and FC-SB-6) a method to differentiate the NVMeoFC usage of FCP from the other protocols is needed. This annex describes the method by which this identification occurs.

F.1.2 T10 definitions

FCP was defined to transport SCSI data frames (see FCP-4). Therefore the original FCP CMND_IU header was defined to contain SCSI structures. The FCP_LUN field in the FCP_CMND IU defined for transporting SCSI commands is shown in table F.1.

Table F.1 - FCP_LUN field in FCP_CMND IU payload

Bit / Byte	7	6	5	4	3	2	1	0
0	FCP_LUN							
7								

The first two words of the FCP_CMND IU consist of the FCP_LUN. This field contains the SCSI LUN (see SAM-6). The format of the SCSI LUN is shown in table F.2.

Table F.2 - SCSI LUN format

Bit / Byte	7	6	5	4	3	2	1	0
n	Address Method (11b)		Length		Extended Address Method			
n+1	(MSB) Extended Address Method Specific							
...								
m								(LSB)

F.1.3 Command IU Identification

If the first byte of a frame with type 08h with information category unsolicited command is set to FDh (i.e., reserved for T11), then the second byte, defined by T11, identifies the protocol which defines the structure of the command IU. For NVMeoFC this value is 28h. The complete table of values is defined in FC-FS-6.